②

TECHNICAL REPORT EL-92-34

DESIGN OF FIXED-FACILITY MULTISPECTRAL CAMOUFLAGE NETTING SUPPORT SYSTEMS

by

James W. Epps, Marion W. Corey

Department of Civil Engineering
Mississippi State University
Mississippi State, Mississippi 39762

October 1992

Final Report

Prepared for DEPARTMENT OF THE ARMY
US Army Corps of Engineers
Washington, DC 20314-1000

Under Contract No. DACA39-90-K-0011

Monitored by Environmental Laboratory
US Army Engineer Waterways Experiment Station
3909 Halls Ferry Road, Vicksburg, Mississippi 39180-6199

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE October 1992	3. REPORT TYPE AND DATES COVERED Final report

4. TITLE AND SUBTITLE Design of Fixed-Facility Multispectral Camouflage Netting Support Systems	5. FUNDING NUMBERS C DACA39-90-K-0011 TA CO WU 026 PR P4A162784AT40
6. AUTHOR(S) James W. Epps Marion W. Corey	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Department of Civil Engineering Mississippi State University, Mississippi State, MS 39762	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) US Army Corps of Engineers, Washington, DC 20314-1000 US Army Engineer Waterways Experiment Station Environmental Laboratory 3909 Halls Ferry Road, Vicksburg, MS 39180-6199	10. SPONSORING/MONITORING AGENCY REPORT NUMBER Technical Report EL-92-34

11. SUPPLEMENTARY NOTES

Available from National Technical Information Services, 5285 Port Royal Road, Springfield, VA 22161.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This study describes a computer model for the design of fixed-facility multispectral camouflage netting support systems. The model was developed in FORTRAN, and the report provides a detailed description of the input, edit, and execution procedures. The model considers the typical tension structures theory, the single cell loads division, support member characteristics, sag distance, cable types, and description of candidate materials with pricing. The model has input/edit routines for pricing data, general site description, and structure geometry. Characterization of the support members, netting, tension members, and anchor system is also established in the model. The report includes the program source codes, an example program output, and the program user guide.

14. SUBJECT TERMS Anchor systems, Camouflage, Multispectral, Support systems, Tension members	15. NUMBER OF PAGES 236
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

Contents

List of Figures

List of Tables

Preface

This study was conducted by personnel of the Department of Civil Engineering at Mississippi State University (MSU), for the U.S. Army Engineer Waterways Experiment Station (WES), under Contract No. DACA39-90-K-0011. The study was conducted as part of Department of the Army Project No. P4A162784AT40, Task CO, Work Unit 026, Fixed-Facility CCD Design & Evaluation Technologies, which is sponsored by Headquarters, U.S. Army Corps of Engineers (HQUSACE).

The report was prepared by Dr. James W. Epps and Dr. Marion W. Corey from the Department of Civil Engineering of MSU. Mr. Bartley P. Durst, Dr. Jonathan C. Duke, Jr., and Mr. Gerardo I. Velazquez of the WES Camouflage Field Demonstration Team assisted in the preparation of the report.

The study was conducted under the general supervision of Dr. John Harrison, Chief, Environmental Laboratory (EL), and Dr. Victor E. LaGarde III, Chief, Environmental Systems Division (ESD), and under the direct supervision of Mr. Malcolm P. Keown, Chief, Environmental Constraint Group (ECG), ESD, and Dr. Jonathan C. Duke, Jr., ECG, Technical Team Leader of the Camouflage Field Demonstration Team. The HQUSACE Technical Monitor was Mr. Mike Shama.

At the time of publication of this report, Director of WES was Dr. Robert W. Whalin. Commander was COL Leonard G. Hassell, EN.

This report should be cited as follows:

Corey, M.W., and Epps, J.W. (1992). "Design of fixed-facility multispectral camouflage netting support systems," Technical Report EL-92-34, prepared by the Department of Civil Engineering, Mississippi State University, for U.S. Army Engineer Waterways Experiment Station, Vicksburg, MS.

Conversion Factors, Non-SI to SI Units of Measurement

Non-SI units of measurement used in this report can be converted to SI units as follows:

Multiply	By	To Obtain
degrees (angle)	0.01745329	radians
feet	0.3048	meters
inches	2.54	centimeters
pounds (force) per square foot	47.88026	pascals
pounds (mass) per cubic foot	16.01846	kilograms per cubic meter

1 Main Program

A computer model for the design of fixed-facility multispectral camouflage netting support systems was developed by the Department of Civil Engineering at Mississippi State University for the U.S. Army Engineer Waterways Experiment Station in Vicksburg, MS. This document is intended to describe the computer model in enough detail that a thorough understanding of the model can be obtained.

The computer model has four major self-contained program options, each of which returns control to the main-line routine when completed. The four program options are as follows: (1) INput, (2) EDit, (3) EXecute, and (4) ENd. The two-character keywords for the program options are stored in the POP() array and are loaded from DATA/READ statements at program initiation. The general logic of the program is illustrated by the flow chart as shown in Figure 1 on the following page.

The initial program activity involves loading the data addresses from a pre-defined array, NAD(), which is defined in the COMMON block. A complete description of the COMMON block and its contents is provided in Chapter 2 Data Addressing and Chapter 3 Utility Routines of this report. Following the loading of the data addresses, the main program option prompt is displayed, and the user is required to provide one of the four program options. If one of the four options is provided, program execution continues. If one of the four options is not provided, an error message is displayed, and control is returned to the program option prompt. The source codes for this input are provided below:

```
100 DIM POP(8) AS STRING*2,BAYS(25)
270 DATA "IN","in","ED","ed","EX","ex","EN","en"
290 FOR I=1 TO 8:READ POP(I):NEXT I
360 REM
370 CLS
430 PRINT:PRINT
440 INPUT "OPTIONS: INPUT, EDIT, EXECUTE OR END (IN/ED/EX/END)";A$
450 FOR IZ=1 TO 8:IF MID$(A$,1,2)=POP(IZ) THEN 480
460 NEXT IZ
470 GOTO 3890
480 IF IZ>6 THEN CLOSE:GOTO 3900
```

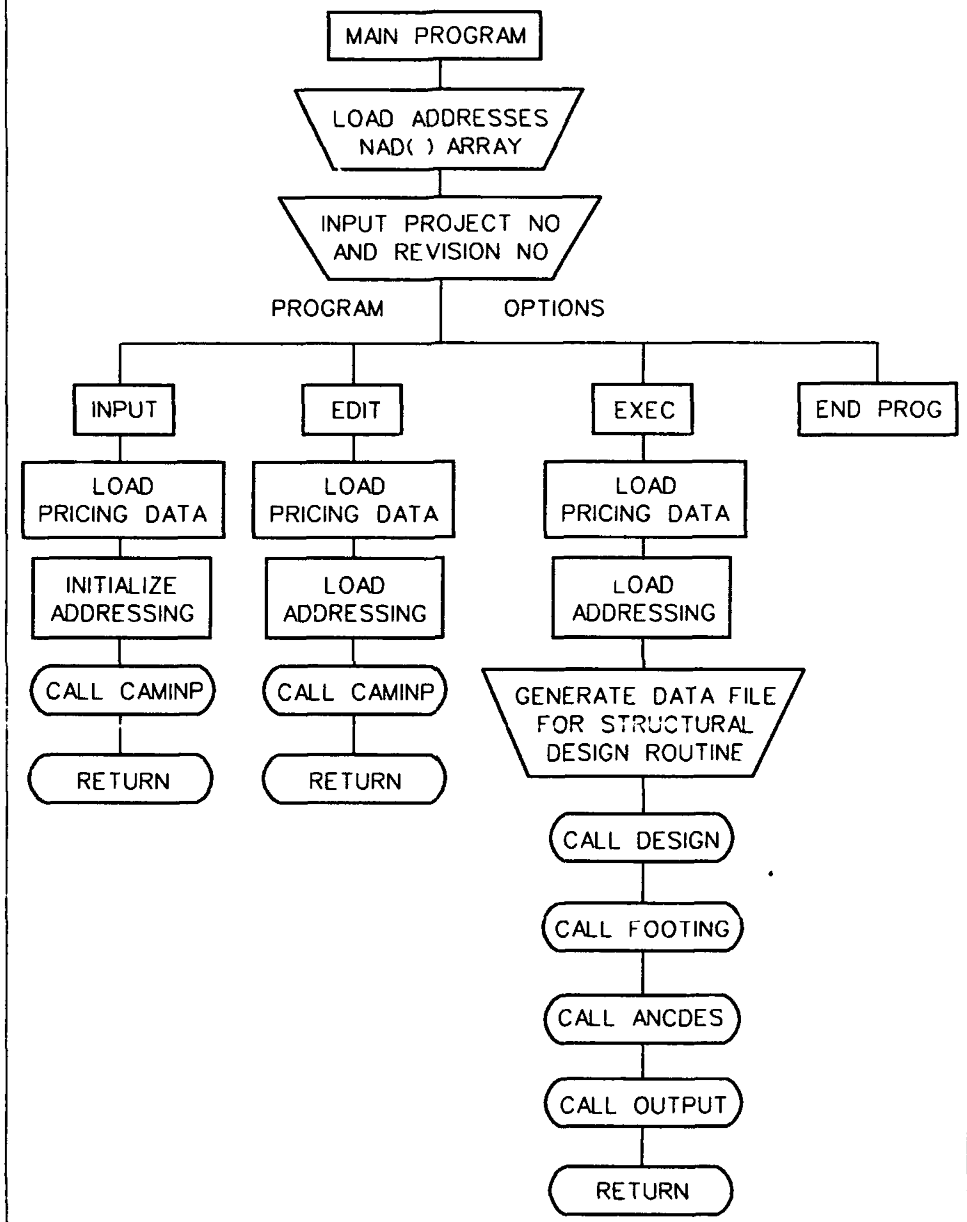

Figure 1. Camouflage main program logic

```
490 REM

3890 PRINT:PRINT A$;" - is an invalid response - try again
     ":GOTO 370
3900 END
```

Following the program option input, the user is required to provide a six-character project name and a one-character revision number. These inputs are used to define the file name to which the project data are to be stored. The project data file name will be as follows:

A:Pxxxxxxy.DAT

where

A:	Data storage will automatically occur on the floppy drive
P	File name prefix
xxxxxx	Six-character project name
y	One-character revision number
DAT	File extension name

This data file will be defined by the program and created during any **INPUT** program option executions. During subsequent program executions, the data file will be accessed automatically. During the initiation of an **INPUT** program operation, the program checks to ensure that the designated file name does not exist on the floppy disk. If the file name does exist on the floppy drive, program operation is automatically switched to an **EDIT** mode to avoid the loss of data. During the initiation of either an **EDIT** or **EXEC** program operation, the program checks to ensure that the designated file name exists on the floppy disk. Program execution is not continued until this condition is satisfied. The source codes that perform these data inputs and create the project .DAT data file are provided below:

```
510 PRINT:INPUT "PROJECT NUMBER (SIX DIGIT MAX) ";ESN$
520 IF ESN$="DIR" OR ESN$="dir" THEN
522       CALL OUTDIR(DUM$()):CLS
524       GOTO 510
526 ENDIF
530 IF LEN(ESN$)>6 THEN ESN$=ESN$+"0":GOTO 530
540 PRINT:INPUT "REVISION NUMBER  (0 - 9) ";RV$
550 FL$="A:P"+MID$(ESN$,1,6)+MID$(RV$,1,1)+".DAT"
```

Input Program Option. When the **INPUT** program option is selected, the program will perform five functions, as follows: (1) load the program pricing data, (2) check to ensure that the specified data file does not exist and initialize the addressing data, (3) open the necessary addressing data files, (4) call the CAMINP routine to perform the data input, and (5) return control to the program option input. Each of these five program functions will be discussed in detail.

Load Pricing Data. The program pricing data are stored in the PRICE.LAT data file on the resident (hard) disk. The PRICE.LAT data file is a random access data file with 130-byte records. The contents of this data file include the individual material items and their structural/ physical characteristics. The contents and organization of this data file are described in Chapter 4 Pricing Data and Chapter 3 Utility Routines of this report. The first data record contains 32 four-byte single precision numeric values which contain the beginning address of material item use codes which begin with a blank, the digits zero through nine, and the alphabetic characters A through U. The second data record contains the four-byte beginning addresses of the remainder of the use codes V through Z and the four-byte number of material data items contained in the file. The use code beginning addresses are loaded in the BEGAD() array, and the number of material data items is located in the NMARK variable. Data records 3-79 each contain 26 five-byte (character) names of the material items stored in the data file. A total of 2,002 material data items are allowed in the data file (76 records with 26 items per record). The five-byte names are stored in the MARK() array, which has been implicitly defined as a five-byte string array in the COMMON. When these data have been loaded, the field lengths of the 18 material data items are loaded into the LNG() array. The source codes to perform the loading of the pricing data are provided below:

```
610 REM INPUT MODE SPECIFIED
620 OPEN DD$+"PRICE.LAT" AS #5 LEN=130:FIELD #5, 130 AS BG$
630 GET #5,1:FOR I=1 TO 32:JZ=(I-1)*4+1
640 BEGAD(I)=CVS(MID$(BG$,JZ,4)):NEXT I
650 GET #5,2:FOR I=33 TO 37:JZ=(I-33)*4+1
660 BEGAD(I)=CVS(MID$(BG$,JZ,4)):NEXT I:NMARK=CVS(MID$(BG$,21,4))
670 FOR I=3 TO 79:GET #5,I:IJ=(I-3)*26:FOR JZ=1 TO 26
680 KZ=(JZ-1)*5+1:MARK(IJ+JZ)=MID$(BG$,KZ,5):NEXT JZ:NEXT I
690 REM
700 FOR MZ=1 TO 18:SM=0:IF MZ=1 THEN 720
710 FOR NZ=1 TO MZ-1:SM=SM+LNG(NZ):NEXT NZ
720 FIELD #5, SM AS DUMM$, LNG(MZ) AS PRD$(MZ):NEXT MZ
730 REM
```

Data File Checking and Addressing Initialization. When the user has input the desired project name and revision number, the associated data file is opened and the first 128-byte record read. The first four bytes of this record contain a single-precision value of the next available record for data storage in the file. If this value is zero, the data file has not been defined, and the data addressing is initialized. The input of project data is then allowed to continue. If this value is not zero, the data file is assumed to exist, and program operation is switched to **EDIT** mode. The source codes to perform the data file checking and addressing initialization are provided below:

```
550 FL$="A:P"+MID$(ESN$,1,6)+MID$(RV$,1,1)+".DAT"
560 OPEN FL$ AS #1 LEN=128:FIELD #1, 128 AS AA$
570 GET #1,1:NXRC=CVS(MID$(AA$,1,4)):CLOSE #1
580 IF NXRC>0 THEN 960   : REM (EDIT OR EXEC MODE ONLY)
600 REM
740 FOR I=1 TO 31:IREC(I)=0:NEXT I:NXRC=2
```

Opening Associated Data Files. Three program data files are necessary for the proper maintenance and storage of the project data. These three data files are as follows: (1) the PROJECT.DAT file as described above, (2) the DATA.LOC data file, and (3) the DATA.FLD data file. The last two data files are described in Chapter 2 Data Addressing of this report. The DATA.LOC data file is a random-access file of four-byte records which contain the single-precision screen locations of the individual data items. The DATA.FLD data file is a random-access file of four-byte records which contain the single-precision field lengths of the individual data items. The source codes to define and open these data files are provided below:

```
750 FL$="P"+MID$(ESN$,1,6)+MID$(RV$,1,1)+".DAT"
760 OPEN DD$+FL$ AS #1 LEN=128:FIELD #1, 128 AS AA$
770 OPEN DD$+"DATA.LOC" AS #2 LEN=4:FIELD #2, 4 AS DL$
780 OPEN DD$+"DATA.FLD" AS #3 LEN=4:FIELD #3, 4 AS DF$
```

Call CAMINP Routine and Return. The CAMINP Routine will allow the sequential input of the project data. A complete description of the organization and function of the routine is provided in Chapter 5 CAMINP Routine of this report. As each input screen is successfully processed, the .DAT data file contents and project addressing are updated. At the conclusion of the input, the complete contents of the .DAT data file are copied from the resident (hard or RAM) disk to the floppy, and control is returned to the program option input.

Edit Program Option. When the **EDIT** program option is selected, the program will perform five functions, as follows: (1) load the program pricing data, (2) check to ensure that the specified data file exists, transfer the project data from the floppy drive to a designated drive (hard disk or RAM drive), and load the addressing data, (3) open the necessary addressing data files, (4) call the CAMINP routine to perform the data editing, and (5) return control to the program option input. Each of these five program functions will be discussed in detail.

Load Pricing Data. The program pricing data are stored in the PRICE.LAT data file on the resident (hard) disk. The PRICE.LAT data file is a random access data file with 130-byte records. The contents of this data file include the individual material items and their structural/physical characteristics. The contents and organization of this data file is described in Chapter 3 Utility Routines and Chapter 4 Pricing Data of this report. The first data record contains 32 four-byte single precision numeric values which contain the beginning address of material item use codes which begin with a blank, the digits zero through nine, and the alphabetic characters A through U. The second data record contains the four-byte beginning addresses of the remainder of the use codes V through Z and the four-byte number of material data items contained in the file. The use code beginning addresses are loaded in the BEGAD() array, and the number of material data items is located in the NMARK variable. Data records 3-79 each contain 26 five-byte (character) names of the material items stored in the data file. A total of 2,002 material data items are allowed in the data file (76 records with 26 items per record). The five-byte names are stored in the MARK() array, which has been implicitly defined as a five-byte string array in the COMMON. When these data have been loaded, the field lengths of the 18 material data items are loaded into the LNG() array. The source codes to perform the loading of the pricing data are provided below:

```
1070 OPEN DD$+"PRICE.LAT" AS #5 LEN=130:FIELD #5, 130 AS BG$
1080 GET #5,1:FOR I=1 TO 32:JZ=(I-1)*4+1
1090 BEGAD(I)=CVS(MID$(BG$,JZ,4)):NEXT I
1100 GET #5,2:FOR I=33 TO 37:JZ=(I-33)*4+1
1110 BEGAD(I)=CVS(MID$(BG$,JZ,4)):NEXT I:NMARK=CVS(MID$(BG$,21,4))
1120 FOR I=3 TO 79:GET #5,I:IJ=(I-3)*26:FOR JZ=1 TO 26
1130 KZ=(JZ-1)*5+1:MARK(IJ+JZ)=MID$(BG$,KZ,5):NEXT JZ:NEXT I
1140 REM
1150 FOR MZ=1 TO 18:SM=0:IF MZ=1 THEN 1170
1160 FOR NZ=1 TO MZ-1:SM=SM+LNG(NZ):NEXT NZ
1170 FIELD #5, SM AS DUMM$, LNG(MZ) AS PRD$(MZ):NEXT MZ
```

Data File Checking, Data Transfer, and Addressing Data Loading. When the user has input the desired project name and revision number, the associated data file is opened and the first 128-byte record read. The first four bytes of this record contain a single-precision value of the next available record for data storage in the file. If this value is zero, the data file has not been defined, and an error message is displayed. If this value is not zero, the data file is assumed to exist, and the contents of the data file are transferred from the floppy drive to the designated disk drive. This designated disk drive may be either the hard disk or a RAM disk, as specified from the contents of the file COMFIL (see Chapter 3 Utility Routines). As this data transfer is being accomplished, the data addressing array IREC() is loaded. This array contains the beginning record number of the data for each input/edit screen that will be displayed during the edit process. The source codes that accomplish these functions are provided below:

```
550 FL$="A:P"+MID$(ESN$,1,6)+MID$(RV$,1,1)+".DAT"
560 OPEN FL$ AS #1 LEN=128:FIELD #1, 128 AS AA$
570 GET #1,1:NXRC=CVS(MID$(AA$,1,4)):CLOSE #1
580 IF NXRC>0 THEN 960   : REM (EDIT OR EXEC MODE ONLY)
```

```
590 IF IZ>2 THEN 820
600 REM
```

The error-handling occurs if the next available record for data storage (NXRC) is zero, and the EDIT program option has been specified (IZ=3 or 4). The specified data file does not exist and an error message is displayed. The user must then repeat the project name and revision number input to satisfy the input requirements of the EDIT program option. The source codes for this program function are provided on the following page:

```
820 COLOR 31,1,1:LOCATE 24,2
830 A$="Project No. \     \  Rev No. \\  does not exist - Press
    any key"
840 PRINT USING A$;ESN$,RV$;
850 A$=INKEY$:IF LEN(A$)=0 THEN 850
860 KILL FL$
870 COLOR 15,1,1:CLS:GOTO 370
```

Once the specified data file has been determined to exist, the data from that file is transferred to the specified disk drive from the floppy, and the data addressing loaded to the IREC() array. The source codes to provide these funtions are provided below:

```
 950 REM
 960 FL$="P"+MID$(ESN$,1,6)+MID$(RV$,1,1)+".DAT"
 970 OPEN DD$+FL$ AS #1 LEN=128:FIELD #1, 128 AS AA$
 980 OPEN "A:"+FL$ AS #2 LEN=128:FIELD #2, 128 AS BA$
 990 GET #2,1:RA$=BA$:NXRC=CVS(MID$(RA$,1,4))
1000 FOR I=1 TO 31:IJ=I*4+1:IREC(I)=CVS(MID$(RA$,IJ,4)):NEXT I
1010 LSET AA$=BA$:PUT #1,1
1020 FOR I=2 TO NXRC-1:GET #2,I:LSET AA$=BA$:PUT #1,I:NEXT
     I:CLOSE #2
1030 REM
```

Opening Associated Data Files. Three program data files are necessary for the proper maintenance and storage of the project data. These three data files are as follows: (1) the project .DAT file as described above, (2) the DATA.LOC data file, and (3) the DATA.FLD data file. The last two data files are described in Chapter 2 Data Addressing of this report. The DATA.LOC data file is a random-access file of four-byte records which contain the single-precision screen locations of the individual data items. The DATA.FLD data file is a random-access file of four-byte records which contain the single-precision field lengths of the individual data items. The source codes to define and open these data files are provided below:

```
1040 OPEN DD$+"DATA.LOC" AS #2 LEN=4:FIELD #2, 4 AS DL$
1050 OPEN DD$+"DATA.FLD" AS #3 LEN=4:FIELD #3, 4 AS DF$
```

Call CAMINP Routine and Return. The CAMINP Routine will allow the sequential editing of the project data. A complete description of the organization and function of the routine is provided in Chapter 5 CAMINP Routine of this report. As each edit screen is successfully processed, the .DAT data file contents and project addressing are updated. At the conclusion of the input, the complete contents of the .DAT data file are copied from the resident (hard or RAM) disk to the floppy, and control is returned to the program option input.

Exec Program Option. When the **EXEC** program option is selected, the program will perform seven functions, as follows: (1) load the program pricing data, (2) check to ensure that the specified data file exists, transfer the project data from the floppy drive to a designated drive (hard disk or RAM drive), and load the addressing data, (3) open the necessary addressing data files, (4) generate the necessary data files for the individual design routines, (5) sequentially call the DESIGN, FOOTING, and ANCDES design routines, (6) call the OUTPUT routine, and (7) return control to the program option input. The first three program functions are the same as for the EDIT program option and will not be repeated. Each of the remaining four program functions will be discussed in detail.

Design Routine Data Generation. The next program function of the EXEC program option is to generate the necessary data for the DESIGN, FOOTING, and ANCDES design routines. Each of these design routines will be discussed in Chapters 6, 7, and 8 respectively. The data generation for each routine, however, follows the same sequence of program operations. These operations are as follows: (1) if necessary, open the required data file to which the project data are to be written, (2) extract the necessary data from the specified project, (3) either write the extracted data to the required data file or pass the extracted data to the design routine directly, and (4) if necessary, close the required data file to which the project data were written. The DESIGN routine is written in FORTRAN and must be SHELLed from the main line routine; therefore, any data transfer must be accomplished through a sequential data file. Both the FOOTING and ANCDES routines are written in BASIC, and the data transfer may be accomplished through the arguments to the CALL statement. The data generation for each routine will be detailed and source codes provided to illustrate the discussions.

DESIGN Routine Data Generation. The DESIGN routine accepts its input from a sequential ASCII data file named DESINPUT.DAT. This data file is opened for output, and sequential write statements will be used to record the data as required by the routine. Because this data file is sequential, the current contents of the file are lost when the file is initially opened and are replaced by the subsequent write statements. The DESIGN routine requires that the DESINPUT.DAT data file contain the following: one record containing the number of bays to be analyzed, followed by an additional record for each bay spacing, followed by eight records containing the remainder of the project data. The generation of each of these data records is described with the associated source codes.

The initial data required involve the bay spacings along the length of the structure. The bay spacing data were input on the second input screen and are located in the project data file beginning at record IREC(2). The variable NAD(2) contains the number of data items contained on the second input screen. The GETFLD routine is called to load the field lengths of the individual data items which are returned in the DUM() array. The DSKRD routine is called to read the individual data items beginning with record number IREC(2) and return the data in the DUM$() string array. Chapter 3 Utility Routines details the use of the GETFLD and DSKRD routines to extract data from the project data file. The DUM$() array will contain 27 (NAD(2)) items when control is returned from the DSKRD routine. The bay spacings are stored in data items 2 through 21 in a format of xx @ xxxx.xx where the even data items (2, 4, 6, 8, ... 20) contain a number of spacings and the odd data items (3, 5, 7, 9 .... 21) contain the bay spacing. Each of the even (DUM$(IZ)) data items is loaded to a dummy string A$ which is then passed to the SCAN routine. The SCAN routine is used to extract the numeric data from the A$ string which contains a single number. This number will be returned in the array V() as the first value in the array (V(1)). If the number is equal to zero, the field is assumed to have been blank, and the next even data field is examined. If the number is greater than zero, the number is loaded to the NBY variable, and the odd field (DUM$(IZ+1)) is loaded to A$ and passed to the SCAN routine. The bay spacing is returned as the first value in the V() array (V(1)). The number of bays variable (NBAY) is incremented by the number of spacings specified (NBY), and the bay spacings array (BAYS()) is loaded with the number of bays spacings specified. This sequence of operations is repeated until all of the even data items from 2 to 21 are extracted. When the data extraction is completed, the variable NBAY contains the number of bays specified, and the variable array BAYS() contains the individual bay spacings. The first record written to the DESINPUT.DAT data file is the number of bay spacings, and an additional record is written for each bay spacing specified. The associated source codes to perform these operations are provided below:

```
1580 OPEN "DESINPUT.DAT" FOR OUTPUT AS #4
1590 REM
1600 REM BAYS AND SPACINGS FROM GEOMETRY DATA
1610 NPR=2:NLOC=NAD(NPR):CALL GETFLD(NPR,DUM())
1620 FREC=IREC(NPR):CALL DSKRD(FREC,NDUM,DUM(),DUM$())
1630 NBAY=0:FOR JZ=2 TO 20 STEP 2
1640 A$=DUM$(JZ):CALL SCAN(A$,V(),NN,B$(),NW):IF V(1)=0 THEN 1670
1650 NBY=V(1):A$=DUM$(JZ+1):CALL SCAN(A$,V(),NN,B$(),NW)
1660 FOR KZ=1 TO NBY:NBAY=NBAY+1:BAYS(NBAY)=V(1):NEXT KZ
1670 NEXT JZ
1680 PRINT #4,USING " ###";NBAY
1690 FOR JZ=1 TO NBAY:PRINT #4,USING " ####.##";BAYS(JZ):NEXT JZ
```

The next data input required by the DESIGN routine involves the width of the camouflage structure, the minimum spacing of supports along the width dimension, height of the support, guy condition of the structure, the maximum sag allowed, and the maximum displacement error allowed. These data were also input on the second input screen and are currently stored in data items 22, 23, 24, 25, 26, and 27 of the DUM$() array. Each numeric data item is extracted using the SCAN routine as previously discussed and stored in a program variable, as follows: structure width from data item 22 in variable SWIDE, minimum spacing from data item 23 in variable SPACE, support height from data item 24 in variable HTPOL, maximum sag from data item 26 in variable SAGMX, and maximum displacement error from data item 27 in variable ALLDSP. The guy condition is from data item 25 and is loaded directly into the string variable GUY$. The next data record written contains the structure width dimension. The source codes for this data extraction and file generation are provided below:

```
1710 REM LOAD REMAINDER OF ITEMS FROM GEOMETRY DATA
1720 A$=DUM$(22):CALL SCAN(A$,V(),NN,B$(),NW):SWIDE=V(1)
1730 A$=DUM$(23):CALL SCAN(A$,V(),NN,B$(),NW):SPACE=V(1)
1740 A$=DUM$(24):CALL SCAN(A$,V(),NN,B$()·,NW):HTPOL=V(1)
1750 GUY$=DUM$(25):IF MID$(GUY$,1,1)="y" THEN GUY$="Y"
1760 IF MID$(GUY$,1,1)="n" THEN GUY$="N"
1770 A$=DUM$(26):CALL SCAN(A$,V(),NN,B$(),NW):SAGMX=V(1)
1775 A$=DUM$(27):CALL SCAN(A$,V(),NN,B$(),NW):ALLDSP=V(1)
1780 PRINT #4,USING " ####.## ";SWIDE
```

The next data input required by the DESIGN routine involves the material characteristics of the specified support members. These data were provided on input screens 3 and 4, with as many as 15 (JZ) individual support members on each screen. With each support member, the following data are provided: (1) a use indicator (Y/N), (2) a five-character name, (3) maximum length, (4) allowable strength, (5) modulus of elasticity, (6) cross-sectional area, (7) moment of inertia, (8) unit cost, and (9) cross-sectional width. The data contained on the input screens are loaded into the DUM$() array, and the use indicator for each item is checked until a 'Y' (or 'y') is found. If no support member was specified on the first input screen (NPR=3), the data are loaded from the second input screen (NPR=4). If no support member was specified on either input screen, an error message is displayed and the execution terminated. This condition should not occur because the CAMINP routine will force one use indicator of 'Y' before input is allowed to continue. Each screen contains up to 135 (NLOC) data items (15 members with 9 characteristics per member). The use indicator (DUM$(IJ+1)) is checked for a 'Y' (or 'y'), and when found, the following data are loaded: (1) member name POLMK$ from DUM$(IJ+2), (2) maximum member length POLEN from DUM$(IJ+3), (3) allowable strength PALLOW from DUM$(IJ+4), (4) modulus of elasticity PMODE from DUM$(IJ+5), (5) cross-sectional area PAREA from DUM$(IJ+6), (6) moment of inertia PMOMI from DUM$(IJ+7), (7) unit cost PCOST from DUM$(IJ+8), (8) cross-sectional width CWIDE from DUM$(IJ+9), and (9) the member description POLDS$ from DUM$(IJ+9). The next required data record containing the structure height, support member modulus of elasticity, moment of inertia, cross-section area, allowable strength, cross-sectional width, and minimum spacing in the width dimension is then written. The source codes for these program operations are provided on the following page:

```
1790 NPR=3
1800 CALL GETFLD(NPR,DUM()):NLOC=NAD(NPR)*15
1810 FOR JZ=2 TO 15:IJ=(JZ-1)*NAD(NPR)
1820 FOR KZ=1 TO NAD(NPR):DUM(IJ+KZ)=DUM(KZ):NEXT KZ:NEXT JZ
1830 FREC=IREC(NPR):CALL DSKRD(FREC,NDUM,DUM(),DUM$())
1840 FOR JZ=1 TO 15:IJ=(JZ-1)*NAD(NPR)
1850 IF DUM$(IJ+1)="Y" OR DUM$(IJ+1)="y" THEN
1860      POLMK$=DUM$(IJ+2)
1870      A$=DUM$(IJ+3):CALL SCAN(A$,V(),NN,B$(),NW):POLEN=V(1)
1880      A$=DUM$(IJ+4):CALL SCAN(A$,V(),NN,B$(),NW):PALLOW=V(1)
1890      A$=DUM$(IJ+5):CALL SCAN(A$,V(),NN,B$(),NW):PMODE=V(1)
1900      A$=DUM$(IJ+6):CALL SCAN(A$,V(),NN,B$(),NW):PAREA=V(1)
1910      A$=DUM$(IJ+7):CALL SCAN(A$,V(),NN,B$(),NW):PMOMI=V(1)
1920      A$=DUM$(IJ+8):CALL SCAN(A$,V(),NN,B$(),NW):PCOST=V(1)
1930      A$=DUM$(IJ+9):CALL SCAN(A$,V(),NN,B$(),NW):CWIDTH=V(1)
1940      POLDS$=DUM$(IJ+9)
1950      GOTO 2080
1960 ENDIF
1970 NEXT JZ
1980 IF NPR=3 THEN NPR=NPR+1:GOTO 1800
1990 REM
2000 REM PROGRAM ERROR - TERMINATE EXECUTION
2010 B$(1)=" PROGRAM ERROR - Support Member Data Error - Terminate
     Execution"
2020 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 2020
2030 LOCATE 24,2:PRINT B$(1);
2040 A$=INKEY$:IF LEN(A$)=0 THEN 2040
2050 COLOR 15,1,1:CLOSE:GOTO 370
2060 REM
2070 REM SUPPORT MEMBER DATA EXTRACTED - CONTINUE WRITING DATA
2080 PRINT #4,USING " ####.## , ";HTPOL;
2090 PRINT #4,USING " ####.## , ";PMODE;
2100 PRINT #4,USING " ####.## , ";PMOMI;
2110 PRINT #4,USING " ####.## , ";PAREA;
2120 PRINT #4,USING " ####.## , ";PALLOW;
2130 PRINT #4,USING " ####.## , ";CWIDTH;
2140 PRINT #4,USING " ####.## ";SPACE
2150 REM
```

The next data input required by the DESIGN routine involves the material characteristics of the specified tension members. These data were provided on input screens 7 and 8, with as many as 15 (JZ) individual tension members on each screen. With each tension member, the following data are provided: (1) a use indicator (Y/N), (2) a five-character name, (3) allowable strength, (4) modulus of elasticity, (5) coefficient of thermal expansion, (6) unit area, (7) unit weight, and (8) unit cost. The data contained on the input screens are loaded into the DUM$() array, and the use indicator for each item is checked until a 'Y' (or 'y') is found. If no support member was specified on the first input screen (NPR=7), the data are loaded from the second input screen (NPR=8). If no tension member was specified on either input screen, an error message is displayed and the execution terminated. This condition should not occur because the CAMINP routine will force one use indicator of 'Y' before input is allowed to continue. Each screen contains up to 135 (NLOC) data items (15 members with 9 characteristics per member). The use indicator (DUM$(IJ+1)) is checked for a 'Y' (or 'y'), and when found, the following data are loaded: (1) member name CABMK$ from DUM$(IJ+2), (2) allowable strength CALLOW from DUM$(IJ+3), (3) modulus of elasticity CMODE from DUM$(IJ+4), (4) coefficient of thermal expansion CTHEX from DUM$(IJ+5), (5) unit area CAREA from DUM$(IJ+6), (6) unit weight CUNWT from DUM$(IJ+7), (7) unit cost CCOST from DUM$(IJ+8), and (8) the member description CABDS$ from DUM$(IJ+9). The next required data record containing the tension member modulus of elasticity, unit area, allowable strength, maximum allowable sag, and unit weight is then written. The source codes for these program operations is provided below:

```
2160 REM EXTRACT DATA FOR TENSION MEMBERS
2170 NPR=7
2180 CALL GETFLD(NPR,DUM()):NLOC=NAD(NPR)*15
2190 FOR JZ=2 TO 15:IJ=(JZ-1)*NAD(NPR)
2200 FOR KZ=1 TO NAD(NPR):DUM(IJ+KZ)=DUM(KZ):NEXT KZ:NEXT JZ
2210 FREC=IREC(NPR):CALL DSKRD(FREC,NDUM,DUM(),DUM$())
2220 FOR JZ=1 TO 15:IJ=(JZ-1)*NAD(NPR)
2230 IF DUM$(IJ+1)="Y" OR DUM$(IJ+1)="y" THEN
2240      CABMK$=DUM$(IJ+2)
2250      A$=DUM$(IJ+3):CALL SCAN(A$,V(),NN,B$(),NW):CALLOW=V(1)
2260      A$=DUM$(IJ+4):CALL SCAN(A$,V(),NN,B$(),NW):CMODE=V(1)
2270      A$=DUM$(IJ+5):CALL SCAN(A$,V(),NN,B$(),NW):CTHEX=V(1)
2280      A$=DUM$(IJ+6):CALL SCAN(A$,V(),NN,B$(),NW):CAREA=V(1)
2290      A$=DUM$(IJ+7):CALL SCAN(A$,V(),NN,B$(),NW):CUNWT=V(1)
2300      A$=DUM$(IJ+8):CALL SCAN(A$,V(),NN,B$(),NW):CCOST=V(1)
2310      CABDS$=DUM$(IJ+9)
2320      GOTO 2420
2330 ENDIF
2340 NEXT JZ
2350 IF NPR=7 THEN NPR=NPR+1:GOTO 2180
2360 REM
2370 REM PROGRAM ERROR - TERMINATE EXECUTION
2380 B$(1)=" PROGRAM ERROR - Tension Member Data Error - Terminate
     Execution"
2390 GOTO 2020
2400 REM
2410 REM TENSION MEMBER DATA EXTRACTED - CONTINUE WRITING DATA
2420 PRINT #4,USING " ####.## , ";CMODE;
2430 PRINT #4,USING " ####.## , ";CAREA;
```

```
2440 PRINT #4,USING " ####.## , ";CALLOW;
2450 PRINT #4,USING " ####.## , ";SAGMX;
2460 PRINT #4,USING " ####.## ";CUNWT
```

The next data input required by the DESIGN routine involves the material characteristics of the specified nets. These data were provided on input screens 5 and 6, with as many as 15 (JZ) individual netting materials on each screen. With each net, the following data are provided: (1) a use indicator (Y/N), (2) a five-character name, (3) unit area, (4) thickness, (5) unit weight, (6) drag coefficient, and (7) unit cost. The data contained on the input screens are loaded into the DUM$() array, and the use indicator for each item is checked until a 'Y' (or 'y') is found. If no support member was specified on the first input screen (NPR=5), the data are loaded from the second input screen (NPR=6). If no netting material was specified on either input screen, an error message is displayed and the execution terminated. This condition should not occur because the CAMINP routine will force one use indicator of 'Y' before input is allowed to continue. Each screen contains up to 135 (NLOC) data items (15 members with 9 characteristics per net). The use indicator (DUM$(IJ+1)) is checked for a 'Y' (or 'y'), and when found, the following data are loaded: (1) netting name NETMK$ from DUM$(IJ+2), (2) unit size NSIZE from DUM$(IJ+3), (3) thickness NTHICK from DUM$(IJ+4), (4) allowable strength NALLOW from DUM$(IJ+5), (5) unit weight NUNWT from DUM$(IJ+6), (6) drag coefficient NDRAG from DUM$(IJ+7), (7) unit cost NCOST from DUM$(IJ+8), and (8) the netting description NETDS$ from DUM$(IJ+9). The source codes that perform the netting data extraction are provided below:

```
2480 REM EXTRACT DATA FOR NETTING
2490 NPR=5
2500 CALL GETFLD(NPR,DUM()):NLOC=NAD(NPR)*15
2510 FOR JZ=2 TO 15:IJ=(JZ-1)*NAD(NPR)
2520 FOR KZ=1 TO NAD(NPR):DUM(IJ+KZ)=DUM(KZ):NEXT KZ:NEXT JZ
2530 FREC=IREC(NPR):CALL DSKRD(FREC,NDUM,DUM(),DUM$())
2540 FOR JZ=1 TO 15:IJ=(JZ-1)*NAD(NPR)
2550 IF DUM$(IJ+1)="Y" OR DUM$(IJ+1)="y" THEN
2560      NETMK$=DUM$(IJ+2)
2570      A$=DUM$(IJ+3):CALL SCAN(A$,V(),NN,B$(),NW):NSIZE=V(1)
2580      A$=DUM$(IJ+4):CALL SCAN(A$,V(),NN,B$(),NW):NTHICK=V(1)
2590      A$=DUM$(IJ+5):CALL SCAN(A$,V(),NN,B$(),NW):NALLOW=V(1)
2600      A$=DUM$(IJ+6):CALL SCAN(A$,V(),NN,B$(),NW):NUNWT=V(1)
2610      A$=DUM$(IJ+7):CALL SCAN(A$,V(),NN,B$(),NW):NDRAG=V(1)
2620      A$=DUM$(IJ+8):CALL SCAN(A$,V(),NN,B$(),NW):NCOST=V(1)
2630      NETDS$=DUM$(IJ+9)
2640      GOTO 2740
2650 ENDIF
2660 NEXT JZ
2670 IF NPR=5 THEN NPR=NPR+1:GOTO 2500
2680 REM
2690 REM PROGRAM ERROR - TERMINATE EXECUTION
```

```
2700 B$(1)=" PROGRAM ERROR - Netting Data Error - Terminate
     Execution"
2710 GOTO 2020
```

The next data input required by the DESIGN routine involves the material characteristics of the specified anchors. These data were provided on input screens 9 and 10, with as many as 15 (JZ) individual anchors on each screen. With each anchor, the following data are provided: (1) a use indicator (Y/N), (2) a five-character name, (3) length, (4) allowable strength, (5) modulus of elasticity, (6) rod diameter, (7) anchor diameter, and (8) unit cost. The data contained on the input screens are loaded into the DUM$() array, and the use indicator for each item is checked until a 'Y' (or 'y') is found. When each anchor specified on the first input screen (NPR=9) has been checked, the data are loaded from the second input screen (NPR=10). Each screen contains up to 135 (NLOC) data items (15 members with 9 characteristics per member). The use indicator (DUM$(IJ+1)) is checked for a 'Y' (or 'y'), and when found, the anchor counter NANC is incremented and the following data are loaded: (1) anchor name ANCMK$(NANC) from DUM$(IJ+2), (2) length ANLEN(NANC) from DUM$(IJ+3), (3) allowable strength ANALL(NANC) from DUM$(IJ+4), (4) helix diameter ANDIA(NANC) from DUM$(IJ+7), (5) unit cost ACOST(NANC) from DUM$(IJ+8), and (6) the anchor description ANCDS$(NANC) from DUM$(IJ+9). These anchor data are stored in a labeled COMMON to facilitate the data transfer to the ANCDES routine. The source codes for these data extractions are provided below:

```
2730 REM EXTRACT DATA FOR ANCHORS
2740 NPR=9:NANC=0
2750 CALL GETFLD(NPR,DUM()):NLOC=NAD(NPR)*15
2760 FOR JZ=2 TO 15:IJ=(JZ-1)*NAD(NPR)
2770 FOR KZ=1 TO NAD(NPR):DUM(IJ+KZ)=DUM(KZ):NEXT KZ:NEXT JZ
2780 FREC=IREC(NPR):CALL DSKRD(FREC,NDUM,DUM(),DUM$())
2790 FOR JZ=1 TO 15:IJ=(JZ-1)*NAD(NPR)
2800 IF DUM$(IJ+1)="Y" OR DUM$(IJ+1)="y" THEN
2810      NANC=NANC+1:ANCMK$(NANC)=DUM$(IJ+2)
2820      A$=DUM$(IJ+3):CALL SCAN(A$,V(),NN,B$(),NW):ANLEN(NANC)=V(1)
2830      A$=DUM$(IJ+4):CALL SCAN(A$,V(),NN,B$(),NW):ANALL(NANC)=V(1)
2840      A$=DUM$(IJ+7):CALL SCAN(A$,V(),NN,B$(),NW):ANDIA(NANC)=V(1)
2850      A$=DUM$(IJ+8):CALL SCAN(A$,V(),NN,B$(),NW):ACOST(NANC)=V(1)
2860      ANCDS$(NANC)=DUM$(IJ+9)
2870 ENDIF
2880 NEXT JZ
2890 IF NPR=9 THEN NPR=NPR+1:GOTO 2750
```

The final data required involve the soil characteristics, design loadings, and weather data. These data were input on the first input screen and are located in the project data file beginning at record IREC(1). The variable NAD(1) contains the number of data items contained on the first input screen. The GETFLD routine is called to load the field lengths of the individual data items which are returned in the DUM() array. The DSKRD routine is called to read the individual data items beginning with record number IREC(1) and return the data in the DUM$() string array. The DUM$() array will contain 11 (NAD(1)) items when control is returned from the DSKRD routine. The soil cohesion is stored in the first data item. The value in DUM$(1) is loaded to a dummy string A$ which is then passed to the SCAN routine. The SCAN routine extracts the numeric value from the A$ string (V(1)) which is then stored in the variable COH. The remaining data needed are extracted in the same way, as follows: soil unit weight from DUM$(2) stored in the variable GAMMA, the soil angle of internal friction in DUM$(3) stored in ANGLE, the diameter of the bored holes in DUM$(4) stored in BORED, the wind speed, average sustained speed, and gusting wind speed in DUM$(7), DUM$(8), and DUM$(9) stored in WINDA, WINDS, and WINDG, and the snow load in DUM$(10) stored in SNOW. When these data are extracted, the maximum wind speed specified is selected for the design wind, and the final data record required by the DESIGN routine is written. This data record contains the soil angle of friction, unit weight of the netting, design snow load, design wind load, maximum allowable displacement error, guy condition, factor of safety (2.0 assumed), and netting drag coefficient. The associated source codes to perform these operations are provided below:

```
2970 NPR=1:CALL GETFLD(NPR,DUM()):NLOC=NAD(NPR)
2980 FREC=IREC(NPR):CALL DSKRD(FREC,NDUM,DUM(),DUM$())
2990 A$=DUM$(1):CALL SCAN(A$,V(),NN,B$(),NW):COH=V(1)
3000 A$=DUM$(2):CALL SCAN(A$,V(),NN,B$(),NW):GAMMA=V(1)
3010 A$=DUM$(3):CALL SCAN(A$,V(),NN,B$(),NW):ANGLE=V(1)
3020 A$=DUM$(4):CALL SCAN(A$,V(),NN,B$(),NW):BORED=V(1)
3030 A$=DUM$(7):CALL SCAN(A$,V(),NN,B$(),NW):WINDA=V(1)
3040 A$=DUM$(8):CALL SCAN(A$,V(),NN,B$(),NW):WINDS=V(1)
3050 A$=DUM$(9):CALL SCAN(A$,V(),NN,B$(),NW):WINDG=V(1)
3060 A$=DUM$(10):CALL SCAN(A$,V(),NN,B$(),NW):SNOW=V(1)
3070 WIND=WINDA:IF WIND<WINDS THEN WIND=WINDS
3080 IF WIND<WINDG THEN WIND=WINDG
3090 REM
3100 REM WRITE REMAINING DATA RECORDS
3110 PRINT #4,USING " ####.## ";ANGLE
3120 PRINT #4,USING " ##.#### , ";NUNWT;
3130 PRINT #4,USING " ####.## , ";SNOW;
3140 PRINT #4,USING " ####.## , ";WIND;
3145 PRINT #4,USING " ##.#### ";ALLDSP
3150 PRINT #4,GUY$
3160 FSAFE=2!:PRINT #4,USING " ####.## ";FSAFE
3170 PRINT #4,USING " ##.#### ";NDRAG
3180 CLOSE
```

Design Routine Calls. When the data extraction is complete, and the required data file for the DESIGN routine has been generated, each of the three design routines are called sequentially. The successful completion of the DESIGN routine is required for the execution of the FOOTING and

ANCDES (anchor) design routines. A sequential scratch data file named DESIGN.DAT is loaded with a zero before the DESIGN routine is executed. When the DESIGN routine executes successfully, a one is loaded into the DESIGN.DAT data file. Before the execution of the FOOTING and ANCDES routines are attempted, the contents of the DESIGN.DAT data file are checked to ensure that the DESIGN routine has been executed successfully. If a zero is read, the program is terminated and no output produced. If a one is read, the design routine executions are allowed to continue. The DESIGN routine is SHELLed to with the input file specified as DESINPUT.DAT. The output of the DESIGN routine is written to a sequential file named DESOUTPT.DAT. Two items of data from this data file are required by the FOOTING design routine: the maximum moment developed at the base of the support members and the maximum tension required in the tension members. These two data items are extracted from the third and fourth lines of the sequential file DESOUTPT.DAT. The file is opened for input, the first two lines are read and discarded, and the next two lines of data are loaded in the dummy string A$ and passed to the SCAN routine. The maximum moment MAXMOM is extracted from the first line of data and converted to foot-pounds. The maximum tension MAXTEN is extracted from the second line of data. The FOOTING routine is called with the required data passed through the arguments of the subroutine CALL statement. If anchor data have been provided (NANC greater than 0) and a guyed condition specified (GUY$=Y), the ANCDES routine is called and the required data passed through the arguments of the subroutine CALL statement. The source codes for these program functions are provided below:

```
3250 OPEN "DESIGN.DAT" FOR OUTPUT AS #1:ZERO=0
3255 PRINT #1, ZERO:CLOSE #1
3260 SHELL "DESIGN < DESINPUT.DAT"
3270 OPEN "DESIGN.DAT" FOR INPUT AS #1:INPUT #1, ZERO:CLOSE #1
3280 IF ZERO<=0 THEN
3290      A$=INKEY$:IF LEN(A$)=0 THEN 3290
3300      GOTO 3800
3310 ENDIF
3320 REM
3330 REM RESULTS FROM DESIGN - DESIGN FOUNDATION FOR POLES
3370 OPEN "DESOUTPT.DAT" FOR INPUT AS #1
3380 LINE INPUT #1, A$:LINE INPUT #1,A$
3390 LINE INPUT #1,A$:CALL SCAN(A$,V(),NN,B$(),NW):MAXMOM=V(1)/12!
3400 LINE INPUT #1,A$:CALL SCAN(A$,V(),NN,B$(),NW):MAXTEN=V(1)
3410 CLOSE
3420 CALL FOOTING(GAMMA,COH,ANGLE,BORED,HTPOL,MAXMOM)
3430 REM
3440 REM CHECK SELECTED ANCHORS FOR MAX GUY TENSION
3450 IF NANC=0 OR GUY$="N" THEN 3530
3460 CALL ANCDES(GAMMA,COH,ANGLE,MAXTEN)
3470 REM
```

OUTPUT Routine. The OUTPUT routine is called when each of the three design routines has successfully completed its output. The output is written to a sequential data file and consists of the following: (1) each of the input screens, (2) graphic description of the final structural design, (3) descriptive summary of the structural design, (4) descriptive summary of the footing design, (5) descriptive summary of the anchoring design, if specified, and (6) a descriptive cost summary for the entire structure. When the output is complete, the user is allowed the opportunity to: (1) browse the output on the display, (2) print the contents of the output file, (3) copy the contents of the output file to floppy disk, or (4) erase the sequential data file from the hard disk. When the output options are completed, the scratch data files (.DAT) are erased from the hard disk, and control is returned to the original program prompt. The source codes that perform the output functions are provided below:

```
3480 REM DESIGN COMPLETE - GRAPHIC PROGRAM OUTPUT
3490 OPEN "DEVICES" FOR INPUT AS #1:NLINE=1
3500 LINE INPUT #1, A$:CALL SCAN(A$,V(),NN,B$(),NW)
3510 IF B$(1)="PRINT" THEN PPORT=V(1):PMODEL=V(2):PSCALE=V(3)
3520 IF B$(1)="SCREEN" THEN SPORT=V(1):SMODEL=V(2):PSCALE=V(3)
3530 IF NLINE=1 THEN NLINE=2:GOTO 3500
3540 CLOSE #1
3550 REM
3560 REM INPUT DATA FOR DRAW PROGRAM
3570 OPEN "CARRY.DAT" FOR OUTPUT AS #1
3580 XPAGE=9!:YPAGE=7!:C$=" #### #### #####.### #####.###"
3590 PRINT #1, USING C$;SPORT,SMODEL,XPAGE,YPAGE
3600 D$=" PROJECT: \     \   REV: \\       CAMOUFLAGE DESIGN PLAN
     VIEW"
3610 PRINT #1, USING D$;ESN$,RV$:CLOSE #1
3620 SHELL "DRAW < CARRY.DAT"
3640 COLOR 15,1,1:CLS:LOCATE 24,2:INPUT "Plot to Printer
     (Y/N) ";A$
3650 IF A$="y" THEN A$="Y"
3660 IF A$="Y" THEN
3670      XPAGE=10!:IF PSCALE>1! THEN XPAGE=13!
3680      YPAGE=7.5:IF PSCALE>1! THEN YPAGE=10!
3690      OPEN "CARRY.DAT" FOR OUTPUT AS #1
3700      PRINT #1, USING C$;PPORT,PMODEL,XPAGE,YPAGE
3710      PRINT #1, USING D$;ESN$,RV$:CLOSE #1
3720      SHELL "DRAW < CARRY.DAT"
3730 ENDIF
3740 KILL "CARRY.DAT"
3750 REM
3760 REM PROGRAM OUTPUT TO DISK FILE
3770 CALL DROUTPUT(ESN$,RV$,DD$,DLNUM,PGVER$)
3780 PL$=LEFT$(FL$,LEN(FL$)-3)+"PRT"
3790 A$="UTILITY "+PL$:SHELL A$
3800 KILL "DESIGN.DAT":KILL FL$
3810 GOTO 370
```

2 Data Addressing

The organization, transfer, and storage of the program data are accomplished by several means. Primary data items are stored in both unlabeled and labeled COMMON statements. This allows the data to be available in subroutines without having to be passed as arguments in the subroutine CALL statements. Primary data storage is accomplished in random access files which are transferred to and from a floppy disk. The contents of this chapter are intended to provide a detailed description of these various data transfer, storage, and retrieval mechanisms. In addition, the organization of the primary project data file and the auxiliary data files needed for the storage/retrieval of these project data is also described. Two library routines, DSKRD and DSKWRT, which are used to retrieve or store the project data are also described.

Unlabeled and Labeled COMMON. The primary data transfer mechanism between both program and library routines is through unlabeled and labeled COMMON. ProBASIC allows the usage of a single unlabeled COMMON block and a maximum of 125 labeled COMMON blocks. When used in conjunction with a DIMension statement, both numeric and string arrays may be stored in both unlabeled and labeled COMMON blocks.

Unlabeled COMMON Block. The unlabeled COMMON block contains the basic numeric and string data required for all phases of program operation. The COMMON block is established at the beginning of each major program module and consists of two DIMension statements followed by the COMMON SHARED statement. The DIMension statements define the maximum space required by the variables included in the COMMON block, and the COMMON SHARED statement defines the order of the individual data in the COMMON block. These three statements must be the first statements of any major program module and are provided below:

```
50 DIM LPTS(135),FLD(135),VAR$(135),IREC(31),NAD(31),PRD$(18)
60 DIM BEGAD(37),MARK(2002) AS STRING*5
70 COMMON SHARED NLOC,NAD(),FLD(),LPTS(),IREC(),
     NMARK,BEGAD(),DL$,DF$,AA$,VAR$(),PRD$(),MARK() AS STRING*5
```

The first three variable arrays, LPTS(), FLD(), and VAR$(), are used in the display of the various input screens during the program operation. The LPTS() array is a single precision numeric array used to store up to 135 values of data field locations on a single screen. The FLD() array is a single precision numeric array used to store up to 135 values of data field lengths on a single screen. The VAR$() array is a string array used to store the input for up to 135 data fields. All data input are stored initially as string data to facilitate the storage and/or retrieval of the data. The IREC() and NAD() arrays are single precision arrays containing the beginning record number of the data in the project data file and number of input data items, respectively, for up to 31 input screens. The PRD$() array is a string array which contains the 18 individual characteristics of each material data item in the pricing data file. The BEGAD() and MARK() arrays are also associated with the pricing data file. The BEGAD() array is a single precision array containing the beginning record number in the pricing data file for each of the 37 acceptable material use codes (blank, numbers 0-9 and characters A-Z). The MARK() array is a string array which contains a maximum of 2,002 five-character names of the material items contained in the pricing data file. A complete description of these variable names with the organization of this data file is presented in Chapter 4 Pricing Data.

The organization of the data in the unlabeled COMMON block is presented in the COMMON SHARED statement. **NLOC** is a single precision variable which contains the total number of data items associated with any one of the input screens. This variable must be defined or calculated before any of the associated program or library subroutines requiring this variable may be used. The **NAD()** array contains the single precision values representing the number of data items associated with a maximum of 31 input screens. The **FLD()** array contains 135 single precision data field lengths. The **LPTS()** array contains 135 single precision data field locations. The **IREC()** array contains single precision values which represent the beginning record number in the project data file for the data contained on each of the 31 input screens. The data in this array are defined during the initial input of the project data and are stored in the first record of the project data file. The first 128-byte record of the project data file will provide the storage of 32 four-byte single precision numbers, and that record will always contain the next available record number for storage (NXRC) and the 31 values of the IREC() array. During the input/edit program operations, these values are defined and/or updated; and, at the conclusion of each input screen, the first data record of the project data file is rewritten. **NMARK** is a single precision value of the number of material data items contained in the pricing data file. The **BEGAD()** array contains 37 single precision values representing the beginning record number in the pricing data file for each of the 37 acceptable material use codes. **DL$** is a string variable to be used only for the extraction of the data field locations. **DF$** is a string variable to be used only for the extraction of the data field lengths. **AA$** is a string variable to be used only for the contents of a single record of data from the project data file. The **VAR$()** array contains up to 135 strings representing the data input on any one of the input screens. The **PRD$()** array contains the 18 characteristics of any one of the material data items contained in the pricing data file. The **MARK()** string array contains the five-character names of a maximum of 2,002 material data items contained in the pricing data file.

Labeled COMMON Blocks. Five labeled COMMON blocks are currently being used by the program. The COMMON SHARED statements and associated DIMension statements are provided below. The GENERAL block contains the variable names that are used throughout the program and is used to simply conserve the memory requirements of the program. The other labeled COMMON blocks, POLES, CABLES, NETS, and ANCHORS, are used to store the critical material data that are specified for the project being currently executed so that these data may be passed to the design and output routines without having to be read from the project data file more than once.

The GENERAL-labeled COMMON contains five DIMensioned variables which are used in all of the program functions. Because these variables are stored in labeled COMMON, they may be passed as the arguments of program library subroutines. Unlabeled COMMON variables may not be passed as the arguments of library subroutines. The **DUM()** array contains 135 single precision values and is used to store the field lengths of the input screen data. The **DUM$()** array contains 135 string values and is used to return the values of the individual data items of an input screen. The **V()** array contains 20 single precision values and is used to store the numeric data which are extracted from an input string using the **SCAN** library routine. The **B$()** array contains 20 string values and is used to store the individual string data items extracted from an input string using the **SCAN** library routine. The **LNG()** array contains the 18 single precision field lengths of the individual characteristics of each material item stored in the pricing data file. The source codes for the GENERAL-labeled COMMON are provided below:

```
80 DIM DUM(135),DUM$(135),V(20),B$(20),LNG(18)
90 COMMON SHARED/GENERAL/DUM(),V(),LNG(),DUM$(),B$(),A$
```

The ANCHORS unlabeled COMMON contains seven variables which contain the material characteristics of a maximum of five anchors to be used in the design routines. The variable **NANC** is the single precision number of anchors whose characteristics are being stored. The **ANCMK$()** array contains five five-character names of the anchors. The **ANLEN()** array contains five single precision lengths (feet) of the anchors. The **ANALL()** array contains five single precision allowable strengths (kips/square inch) of the anchors. The **ANDIA()** array contains five single precision helix diameters (inches) of the anchors. The **ACOST()** array contains five single precision unit costs (dollars/each) of the anchors. The **ANCDS$()** array contains five 15-character descriptions of the anchors. The source codes for the ANCHORS-labeled COMMON are provided below:

```
100 DIM ANCMK$(5),ANLEN(5),ANALL(5),ANDIA(5),ACOST(5),ANCDS$(5)
110 COMMON SHARED/ANCHORS/NANC,ANCMK$(),ANLEN(),ANALL(),
      ANDIA(),ACOST(),ANCDS$()
```

The remaining three labeled COMMON blocks are used to store the material data for a single support member (pole), a single tension member (cable), and a single netting material that will be analyzed in the design routines. The POLES-labeled COMMON contains **POLEN**, the length of the selected support member (feet); **PCOST**, the unit cost of the support member (dollars/each); **POLMK$**, the five-character name of the support member; and **POLDS$**, a 15-character description of the support member. The CABLES-labeled COMMON contains **CCOST**, the unit cost of the tension member (dollars/foot); **CABMK$**, the five-character name of the tension member; and **CABDS$**, a 15-character description of the tension member. The NETS-labeled COMMON contains **NSIZE**, the unit size of each piece of netting (square feet); **NCOST**, the unit cost of the netting (dollars/square foot); **NETMK$**, the five-character name of the netting; and **NETDS$**, a 15-character description of the netting. The source codes for these three labeled COMMON blocks are provided below:

```
120 COMMON SHARED/POLES/POLEN,PCOST,POLMK$,POLDS$
130 COMMON SHARED/CABLES/CCOST,CABMK$,CABDS$
140 COMMON SHARED/NETS/NSIZE,NCOST,NETMK$,NETDS$
```

Auxiliary Data Files. Three auxiliary data files are needed for the program data storage/retrieval functions. These three data files are random access files and are named DATA.ADD, DATA.LOC, and DATA.FLD. These three data files must exist on the specified (DD$) disk drive. These data files can be created by executing the **CAMOBLDR** program (source codes provided at the end of this chapter). The contents, organization, and use of each of the three data files are described below.

DATA.ADD Data File. The DATA.ADD file is a random access data file which contains a single 128-byte record. This 128-byte record can contain up to 32 four-byte single-precision numbers. The current version of the Camouflage Program allows a maximum of 31 values which are the number of data items contained on 31 input screens. The current version of the program contains 10 input screens; therefore, the first 10 data values are defined, and the remaining 21 data items are zero. The current 10 numeric values are 11, 27, 9, 9, 9, 9, 9, 9, 9, and 9, representing 11 data items on the first input screen, 27 data items on the second input screen, and so on to 9 data items on the tenth input screen. The total of these values (110) represents the total number of data items that could be provided for any project input. At the beginning of each major program module, this file is opened, 31 numeric values are extracted from the first record, and these values are stored in the NAD() array. The NAD() array is listed in the blank COMMON statement and is therefore available to every subroutine in the program module. A sample of the source codes which accomplish the loading of these data are provided below:

```
330 OPEN DD$+"DATA.ADD" AS #1 LEN=128:FIELD #1, 128 AS DA$
340 GET #1,1:RA$=DA$:FOR I=1 TO 31:IJ=(I-1)*4+1
350 NAD(I)=CVS(MID$(RA$,IJ,4)):NEXT I:CLOSE #1
```

From the source codes shown above, the file is OPENed and identified as a random access data file with 128-byte records. The first record is read, stored in the string variable DA$, and transferred to the string variable RA$. From that string, the individual numeric data values are extracted from each four-byte section of the string. The variable counter I is used to calculate the beginning character (IJ) of each four-byte string. The numeric value is extracted by the CVS (convert to single precision number) function and stored in the Ith location in the NAD() array.

DATA.LOC Data File. The DATA.LOC file is a random access data file which contains four-byte records. Each four-byte record contains a single precision numeric value which represents the screen location of a particular data item. The screen location is a four-digit number in which the first two digits represent the row (01-24) and the last two digits represent the column (01-80) for the beginning of the data field for the data item. For example, a data field which begins at a screen location of row 16 and column 46 would be indicated by a numeric value of 1646. The beginning location for every data field on each of the ten input screens is stored on sequential four-byte records in the data file. For example, the first input screen provides for 11 data items, so the first 11 values of this data file contain the beginning field locations for these 11 data items. The next 27 values contain the beginning field locations for the 27 data items which can be input on screen number two, and so on to the last nine values are the beginning data fields for the nine data items on input screen number 10. For the current version of the program, the contents of this data file are 110 four-byte single precision values.

Before any of the ten input screens may be displayed, the beginning locations of the data fields on the input screen to be displayed must be extracted from the DATA.LOC data file. These data are extracted with the **GETLOC** subroutine which is available in the program library (see Chapter 3 on Utility Routines). The **GETLOC** subroutine source codes are provided below.

```
190 SUB GETLOC(IP,DUM(1))
200 NST=0:IF IP=1 THEN 220
210 FOR I=1 TO IP-1:NST=NST+NAD(I):NEXT I
220 IEN=NST+NAD(IP):NO=0:FOR I=NST+1 TO IEN:NO=NO+1
230 GET #2,I:DUM(NO)=CVS(DL$):NEXT I
240 END SUB
```

The **GETLOC** subroutine has two arguments: the number of the screen to be displayed (IP) and an array in which the screen locations are returned (DUM()). The first record to be read is calculated by summing the number of data items (NAD()) associated with the preceding input screens (IP-1). The variable NST contains this sum and represents the record number containing the field location of the last data item on the input screen immediately preceding input screen number IP. The data field locations for input screen number IP are located in record numbers NST+1 through NST+NAD(IP). Each four-byte record is read, and the numeric value is extracted by the CVS function and stored in the proper address (NO) of the DUM() array.

Once the field location data are returned from the **GETLOC** subroutine, the contents of the argument array (DUM()) can be transferred to the blank COMMON array, LPTS(), and used by any other program or library routine. In the case where the screen locations are repetitive (the column locations are the same for each row of input), only the first row of values are provided in the NAD() array, and the remainder of the screen locations are calculated. Examples of the source codes which accomplish the data acquisition for both cases are provided below:

```
1130 KEYZ=0:NPR=2:NLOC=NAD(NPR):CALL GETLOC(NPR,DUM())
1140 FOR I=1 TO NAD(NPR):LPTS(I)=DUM(I):NEXT I
```

In the first example, the field locations for the second input screen (NPR=2) are desired. The NAD(2) value is 27, and the total number of data fields (NLOC) is also 27. The **GETLOC** subroutine is called, and 27 values are returned in the DUM() array. These 27 values are then stored in the LPTS() array.

```
2080 KEYZ=0:NPR=3:NYES=0
2090 CALL GETLOC(NPR,DUM()):NLOC=NAD(NPR)*15
2100 FOR I=1 TO NAD(NPR):LPTS(I)=DUM(I):NEXT I
2120 FOR I=2 TO 15:IJ=(I-1)*NAD(NPR):FOR J=1 TO NAD(NPR)
2130 LPTS(IJ+J)=LPTS(J)+100*(I-1)
2140 NEXT J:NEXT I
```

In the second example, the field locations for the third input screen (NPR=3) are desired. The NAD(3) value is 9, which for this case represents nine data items per line. A total of 15 lines of input are allowed, which results in a total number of data items (NLOC) of 135 (NAD(NPR)*15). The **GETLOC** subroutine is called, and the nine screen locations for the first row are returned in the DUM() array. These nine values are then stored in the LPTS() array. The field locations for the remaining 14 rows are calculated by adding 100 to the corresponding field location of the preceding row.

DATA.FLD Data File. The DATA.FLD file is a random access data file which contains four-byte records. Each four-byte record contains a single precision numeric value which represents the length of the input field for a particular data item. The field length for every data item on each of the ten input screens is stored on sequential four-byte records in the data file. For example, the first input screen provides for 11 data items, so the first 11 values of this data file contain the field lengths for these 11 data items. (The next value contains the field lengths for the 27 data items which can be input on screen number two, and so on to the last nine values are the field lengths for the nine data items on input screen number 10.) For the current version of the program, the contents of this data file are 110 four-byte single precision values.

Before any of the ten input screens may be displayed, the field lengths of each data item on the input screen to be displayed must be extracted from the DATA.FLD data file. These data are extracted with the **GETFLD** subroutine which is available in the program library (see Chapter 3 on Utility Routines). The **GETFLD** subroutine source codes are provided below.

```
250 SUB GETFLD(IP,DUM(1))
260 NST=0:IF IP=1 THEN 280
270 FOR I=1 TO IP-1:NST=NST+NAD(I):NEXT I
280 IEN=NST+NAD(IP):NO=0:FOR I=NST+1 TO IEN:NO=NO+1
290 GET #3,I:DUM(NO)=CVS(DF$):NEXT I
300 END SUB
```

The **GETFLD** subroutine has two arguments: the number of the screen to be displayed (IP) and an array in which the screen field lengths are returned (DUM()). The first record to be read is calculated by summing the number of data items (NAD()) associated with the preceding input screens (IP-1). The variable NST contains this sum and represents the record number containing the field length of the last data item on the input screen immediately preceding input screen number IP. The data field lengths for input screen number IP are located in record numbers NST+1 through NST+NAD(IP). Each four-byte record is read, and the numeric value is extracted by the CVS function and stored in the proper address (NO) of the DUM() array.

Once the field length data are returned from the **GETFLD** routine, the contents of the argument array (DUM()) can be transferred to the blank COMMON array, FLD(), and used by any other program or library routine. In the case where the screen field lengths are repetitive (the field lengths are the same for each row of input), only the first row of values are provided in the FLD() array, and the remainder of the screen field lengths are duplicated from the first row of values. Examples of the source codes which accomplish the data acquisition for both cases are provided below:

```
1130 KEYZ=0:NPR=2:NLOC=NAD(NPR)
1150 CALL GETFLD(NPR,DUM())
1160 FOR I=1 TO NAD(NPR):FLD(I)=DUM(I):NEXT I
```

In the first example, the field locations for the second input screen (NPR=2) are desired. The NAD(2) value is 27, and the total number of data fields (NLOC) is also 27. The **GETLOC** subroutine is called, and 27 values are returned in the DUM() array. These 27 values are then stored in the FLD() array.

```
2080 KEYZ=0:NPR=3:NYES=0
2090 NLOC=NAD(NPR)*15
2110 CALL GETFLD(NPR,DUM())
2115 FOR I=1 TO NAD(NPR):FLD(I)=DUM(I):NEXT I
2120 FOR I=2 TO 15:IJ=(I-1)*NAD(NPR):FOR J=1 TO NAD(NPR)
2130 DUM(IJ+J)=DUM(J):FLD(IJ+J)=DUM(J)
2140 NEXT J:NEXT I
```

In the second example, the field lengths for the third input screen (NPR=3) are desired. The NAD(3) value is 9, which for this case represents nine data items per line. A total of 15 lines of input are allowed which results in a total number of data items (NLOC) of 135 (NAD(NPR)*15). The **GETFLD** subroutine is called, and the nine screen field lengths for the first row are returned in the DUM() array. These nine values are then stored in the FLD() array. The field lengths for the remaining 14 rows are simply duplicated from the first row.

Project Data File Organization. The input data for a project are stored in a single random access file containing 128-byte records. The data file name will be composed of the project name and revision number as follows:

PxxxxxxY.DAT

where

P	File name prefix character
xxxxxx	Six-character project name
Y	One-character revision number
DAT	File extension name

Before any program function is attempted, the user will be required to input the desired project name and revision number. For an INput program function, the data file is created on the specified (DD$) disk drive. For any other program function, the data file is assumed to exist on the "A" (floppy) disk drive, and the contents of the file are copied from the floppy disk to the specified disk drive. At the conclusion of each program function, the contents of the data file are copied from the specified disk drive to the floppy disk drive, and the file on the specified disk drive is erased (KILLed).

The data from each input screen are written in the project data file always beginning on a record boundary. In addition, no individual data item is "split" between two consecutive records. If the remaining space of the current record is not sufficient for an individual data item, the current record is written, and that data item is stored at the beginning of the next record. This method of storing the data always results in "unused" space at the end of the 128-byte records, resulting in a larger file size. Data retrieval, however, is greatly facilitated by storing the data in this manner. The first record of each project data file will always contain the same data. The next available record number for writing, NXRC, is a single precision value stored in the first four bytes of that first record. The remaining 124 bytes of the record contain the 31 four-byte single precision values of the IREC() array, the beginning record number of the data for up to 31 input screens of data. The data for the first record are initialized during an INput program function and are updated and written in the first record after each input screen is successfully completed. The remaining records of the project data file contain the individual data items from each input screen, each beginning on a record boundary. An example of how these data are updated and written to the data file is illustrated by the source codes provided below:

```
940 FREC=IREC(NPR):IF FREC=0 THEN FREC=NXRC
950 CALL DSKWRT(FREC)
955 IF IREC(NPR)=0 THEN IREC(NPR)=NXRC:NXRC=FREC+1
960 A$=MKS$(NXRC):FOR I=1 TO 31:A$=A$+MKS$(IREC(I)):NEXT I
970 LSET AA$=A$:PUT #1,1
```

The beginning record number for the current input screen (NPR) is loaded from the IREC() array. If no data have been previously input for the current screen, the corresponding value in the IREC() array is zero and the "next available record" (NXRC) is loaded as the first record number to be written (FREC). The DSKWRT subroutine is called to write the individual data items from the current screen to the project data file beginning at record number FREC. If more than one record is required for the screen data, the current record count will be maintained in FREC by the DSKWRT routine. When the data have been written to the project data file, control is returned from the DSKWRT routine, and FREC will contain the number of the last record written. If the data were written for the first time, the beginning record number in the IREC() array would be zero. If this is the case, IREC(NPR) is loaded with NXRC, the beginning record number of the current data and NXRC is then updated to FREC+1, the next available record number for writing. If the data had previously been written, and had been simply updated, neither of these values would be changed. After these updates have been accomplished, the first record is updated with the current value of NXRC and the current contents of the IREC() array. It is then rewritten to the project data file. A description of the DSKWRT routine follows and should provide a more detailed description of the project data file contents.

DSKWRT Routine. The DSKWRT routine resides in the program library of utility routines. As the input for each screen is successfully completed, this routine is used to store the individual data items from the screen to the project data file and update the current record being written. The source codes for the DSKWRT routine are provided and described below:

```
310 SUB DSKWRT(FREC)
320 RA$=MKS$(NLOC):LGR=4
330 FOR I=1 TO NLOC:NA=FLD(I)-LEN(VAR$(I)):IF NA<=0 THEN 350
340 FOR J=1 TO NA:VAR$(I)=VAR$(I)+" ":NEXT J
350 IF LGR+FLD(I)>128 THEN 370
360 LGR=LGR+FLD(I):RA$=RA$+MID$(VAR$(I),1,FLD(I)):GOTO 410
370 LAD=128-LEN(RA$):IF LAD=0 THEN 390
380 FOR J=1 TO LAD:RA$=RA$+" ":NEXT J
390 LSET AA$=RA$:PUT #1,FREC:FREC=FREC+1
400 RA$=MID$(VAR$(I),1,FLD(I)):LGR=FLD(I)
410 NEXT I:IF LGR=0 THEN 430
420 LAD=128-LGR:FOR I=1 TO LAD:RA$=RA$+" ":NEXT I
425 LSET AA$=RA$:PUT #1,FREC
430 END SUB
```

When control is initially passed to the subroutine, the number of the first record to be written is in FREC. NLOC contains the number of data items to be written, and the FLD() array contains the field length of each of the data items. The individual data items are in the VAR$() array. The total number of data items (NLOC) written is loaded into the first four bytes of the first record to be written (RA$), and the current length (LGR) of the record is set to 4. Each data item (VAR$(I)) is then checked to ensure that its field length is correct. This function must be done because the user may have only used a portion of the field length to input a particular data item. If this is the case, the remaining portion of the field length is filled with blanks.

The current record is checked to ensure that a sufficient number of bytes remain in the RA$ string to store the current data item. If sufficient space exists (LGR+FLD(I)<=128), the current data item is simply added to the RA$ string, the length of the RA$ string is increased by the data item field length (LGR+FLD(I)), and processing of the next data item is begun. If sufficient space does not exist (LGR+FLD(I)>128), the remaining bytes of the current string (128-LEN(RA$)) are filled with blanks, the current string RA$ is loaded (LSET) into the reserved string AA$, and the current record is written to the project data file as record number FREC. FREC is then updated by one (FREC=FREC+1), the current data item is loaded into the first FLD(I) bytes of the RA$ string, and the current length of the RA$ string is set to FLD(I). Processing of the next data item is then ready to begin.

When all of the individual data items have been processed, a check is made to ensure that the last record has been written. If the length of the RA$ string LGR is zero, the last data record has been written, and control is returned to the CALLing routine. If the length of the RA$ string LGR is not zero, a record has been partially filled, but not yet written. The remaining length of the string RA$ (128-LGR) is filled with blanks, the string RA$ loaded (LSET) into the reserved string AA$, and the record is written to the project data file as record number FREC. Control is then returned to the CALLing routine.

DSKRD Routine. The DSKRD routine resides in the program library of utility routines. Before the input/edit program function for each screen is begun, this routine is used to retrieve the individual data items from the project data file. In addition, this routine is also used to retrieve the project data necessary for the execution of the design routines. The source codes for the DSKWRT routine are provided and described below:

```
440 SUB DSKRD (FREC,LOCN,DUM(1),DUM$(1))
450 LGR=4:GET #1,FREC:RA$=AA$:LOCN=CVS(MID$(RA$,1,4))
460 FOR I=1 TO LOCN:IF LGR+DUM(I)>128 THEN 480
470 DUM$(I)=MID$(RA$,LGR+1,DUM(I)):LGR=LGR+DUM(I):GOTO 490
480 FREC=FREC+1:GET #1,FREC:RA$=AA$:LGR=0:GOTO 470
490 NEXT I
500 END SUB
```

The arguments that must be defined before the routine can be called are the beginning record to be retrieved (FREC) and the field lengths of the individual data items (DUM() array). The first record is read and loaded into the string RA$. The number of data items to be retrieved (LOCN) is extracted from the first four bytes of the record, and the number of bytes retrieved from the current record LGR is set to four. The individual data items are stored in the DUM$() string array. As each data item is processed, the remaining number of bytes in the current record is checked to ensure that sufficient space is left for the current data item. If a sufficient number of bytes remain (LGR+DUM(I)<=128), the current data item is extracted from the RA$ from byte LGR+1 for DUM(I) bytes, the extracted value is loaded into the string array DUM$(I), the number of bytes extracted is incremented by the data item field length (LGR+DUM(I)), and processing of the next data item is begun. If sufficient space does not exist in the current record for the data item (LGR+DUM(I)>128), the current record number (FREC) is incremented by one, the next record is read and loaded into the string RA$, and the current number of bytes extracted LGR is set to zero. The data item is extracted from the first DUM(I) bytes of that record and stored in the DUM$(I) string array, the current number of bytes extracted LGR is incremented by the data item field length, and processing of the next data item is begun. When all (LOCN) of the data items have been extracted, control is returned to the CALLing routine.

Addressing Data Summary. The basic data for the each of the ten current input screens is provided below. The arrangement of these data is presented in the order that the screens would appear during an input/edit program function. For each input screen, the following data are provided: (a) a description of the screen, (b) the number of addressing data items (NAD()), (c) the total number of data items (NLOC), (d) the total number of bytes required for storage in the project data file, including four bytes for the total number of data items, and (e) the number of records required to write the data in the project data file. For each data item listed, the screen field location, field length, and a basic description are also provided. After the contents of each screen are described, the source codes for the **CAMOBLDR** program are provided. This stand-alone executable is used to generate the data records for the DATA.ADD, DATA.LOC, and DATA.FLD random access data files.

Input Screen Number 1

Description: Site Specific Project Data

Number of Addressir· Data Items: 11

Total Number of Data Items: 11

Total Number of Bytes Required: 129

Number of Records Required: 2

Data Item	Screen Location	Field Length	Description
1	636	10	Soil cohesion (p:f)
2	739	10	Soil unit weight (pcf)
3	858	10	Angle of internal friction (degs)
4	965	10	Boring diameter (inches)
5	1354	10	Average daily rain (in/day)
6	1457	10	Average daily temperature (degs F)
7	1539	10	Average wind speec (mph)
8	1650	10	Sustained average wind speed (mph)
9	1744	10	Gusting wind speed (mph)
10	1854	10	Ice/snow loading (psf)
11	2154	25	General terrain description

Input Screen Number 2

Description: Camouflage Structure Ger metry

Number of Addressing Data Items: 27

Total Number of Data Items: 27

Total Number of Bytes Required:

Number of Records Required:

Data Item	Screen Location	Field Length	Description
1	648	10	Total structure length (feet)
2	927	2	Number of bay spacings
3	932	4	Bay spacing (feet)
4	937	2	Number of bay spacings
5	942	4	Bay spacing (feet)
6	947	2	Number of bay spacings
7	952	4	Bay spacing (feet)
8	957	2	Number of bay spacings
9	962	4	Bay spacing (feet)
10	967	2	Number of bay spacings
11	972	4	Bay spacing (feet)
12	1127	2	Number of bay spacings
13	1132	4	Bay spacing (feet)
14	1137	2	Number of bay spacings
15	1142	4	Bay spacing (feet)
16	1147	2	Number of bay spacings
17	1152	4	Bay spacing (feet)
18	1157	2	Number of bay spacings
19	1162	4	Bay spacing (feet)
20	1167	2	Number of bay spacings
21	1172	4	Bay spacing (feet)
22	1447	10	Total structure width (feet)
23	1549	10	Minimum pole spacing (feet)
24	1962	10	Exterior pole height (feet)
25	2048	2	Exterior poles guyed (Y/N)
26	2150	10	Maximum allowable cable sag (feet)
27	2260	10	Allowable displacement error (inch)

Input Screen Number 3

Description: Support Member Material Characteristics

Number of Addressing Data Items: 9

Total Number of Data Items: 135[1]

Total Number of Bytes Required: 1054

Number of Records Required: 9

Data Item	Screen Location	Field Length	Description
1	902	1	Use indicator (Y/N)
2	904	6	Item name
3	911	8	Total length (feet)
4	920	8	Allowable strength (ksi)
5	929	8	Modulus of elasticity (ksi)
6	938	8	Cross-sectional area (sqin)
7	947	8	Moment of inertia (in^4)
8	956	8	Unit price ($/each)
9	965	15	Material description

[1] NOTE: Each screen contains 15 rows of data with 9 data items per row. The screen locations and field elngths shown are for the first row. Screen locations for rows 2 through 15 are calculated by adding 100 to the screen location of the previous row for each of the data items.

Input Screen Number 4

Description: Support Member Material Characteristics

Number of Addressing Data Items: 9

Total Number of Data Items: 135[1]

Total Number of Bytes Required: 1054

Number of Records Required: 9

Data Item	Screen Location	Field Length	Description
1	902	1	Use indicator (Y/N)
2	904	6	Item name
3	911	8	Total length (feet)
4	920	8	Allowable strength (ksi)
5	929	8	Modulus of elasticity (ksi)
6	938	8	Cross-sectional area (sqin)
7	947	8	Moment of inertia (in^4)
8	956	8	Unit price ($/each)
9	965	15	Material description

[1] NOTE: Each screen contains 15 rows of data with 9 data items per row. The screen locations and field lengths shown are for the first row. Screen locations for rows 2 through 15 are calculated by adding 100 to the screen location of the previous row for each of the data items.

Input Screen Number 5

Description: Netting Material Characteristics

Number of Addressing Data Items: 9

Total Number of Data Items: 135[1]

Total Number of Bytes Required: 1054

Number of Records Required: 9

Data Item	Screen Location	Field Length	Description
1	902	1	Use indicator (Y/N)
2	904	6	Item name
3	911	8	Unit size (sqft)
4	920	8	Netting thickness (inches)
5	929	8	Allowable strength (ksi)
6	938	8	Unit weight (lbs/sqft)
7	947	8	Drag coefficient
8	956	8	Unit price ($/sqft)
9	965	15	Material description

[1] NOTE: Each screen contains 15 rows of data with 9 data items per row. The screen locations and field lengths shown are for the first row. Screen locations for rows 2 through 15 are calculated by adding 100 to the screen location of the previous row for each of the data items.

Input Screen Number 6

Description: Netting Material Characteristics

Number of Addressing Data Items: 9

Total Number of Data Items: 135[1]

Total Number of Bytes Required: 1054

Number of Records Required: 9

Data Item	Screen Location	Field Length	Description
1	902	1	Use indicator (Y/N)
2	904	6	Item name
3	911	8	Unit size (sqft)
4	920	8	Netting thickness (inches)
5	929	8	Allowable strength (ksi)
6	938	8	Unit weight (lbs/sqft)
7	947	8	Drag coefficient
8	956	8	Unit price ($/sqft)
9	965	15	Material description

[1] NOTE: Each screen contains 15 rows of data with 9 data items per row. The screen locations and field lengths shown are for the first row. Screen locations for rows 2 through 15 are calculated by adding 100 to the screen location of the previous row for each of the data items.

Input Screen Number 7

Description: Tension Member Material Characteristics

Number of Addressing Data Items: 9

Total Number of Data Items: 135[1]

Total Number of Bytes Required: 1054

Number of Records Required: 9

Data Item	Screen Location	Field Length	Description
1	902	1	Use indicator (Y/N)
2	904	6	Item name
3	911	8	Allowable strength (kips)
4	920	8	Modulus of elasticity (ksi)
5	929	8	Coefficient of thermal expansion
6	938	8	Cross-sectional area (sqin)
7	947	8	Unit weight (lbs/ft)
8	956	8	Unit price ($/foot)
9	965	15	*Material description*

[1] NOTE: Each screen contains 15 rows of data with 9 data items per row. The screen locations and field lengths shown are for the first row. Screen locations for rows 2 through 15 are calculated by adding 100 to the screen location of the previous row for each of the data items.

Input Screen Number 8

Description: Tension Member Material Characteristics

Number of Addressing Data Items: 9

Total Number of Data Items: 135[1]

Total Number of Bytes Required: 1054

Number of Records Required: 9

Data Item	Screen Location	Field Length	Description
1	902	1	Use indicator (Y/N)
2	904	6	Item name
3	911	8	Allowable strength (kips)
4	920	8	Modulus of elasticity (ksi)
5	929	8	Coefficient of thermal expansion
6	938	8	Cross-sectional area (sqin)
7	947	8	Unit weight (lbs/ft)
8	956	8	Unit price ($/foot)
9	965	15	Material description

[1] NOTE: Each screen contains 15 rows of data with 9 data items per row.
The screen locations and field lengths shown are for the first row.
Screen locations for rows 2 through 15 are calculated by adding 100 to the screen location of the previous row for each of the data items.

Input Screen Number 9

Description: Anchor Material Characteristics

Number of Addressing Data Items: 9

Total Number of Data Items: 135[1]

Total Number of Bytes Required: 1054

Number of Records Required: 9

Data Item	Screen Location	Field Length	Description
1	902	1	Use indicator (Y/N)
2	904	6	Item name
3	911	8	Rod length (inches)
4	920	8	Allowable strength (ksi)
5	929	8	Modulus of elasticity (ksi)
6	938	8	Rod diameter (inches)
7	947	8	Anchor diameter (inches)
8	956	8	Unit price ($/each)
9	965	15	Material description

[1] NOTE: Each screen contains 15 rows of data with 9 data items per row. The screen locations and field lengths shown are for the first row. Screen locations for rows 2 through 15 are calculated by adding 100 to the screen location of the previous row for each of the data items.

Input Screen Number 10

Description: Anchor Material Characteristics

Number of Addressing Data Items: 9

Total Number of Data Items: 135[1]

Total Number of Bytes Required: 1054

Number of Records Required: 9

Data Item	Screen Location	Field Length	Description
1	902	1	Use indicator (Y/N)
2	904	6	Item name
3	911	8	Rod length (inches)
4	920	8	Allowable strength (ksi)
5	929	8	Modulus of elasticity (ksi)
6	938	8	Rod diameter (inches)
7	947	8	Anchor diameter (inches)
8	956	8	Unit price ($/each)
9	965	15	Material description

[1] NOTE: Each screen contains 15 rows of data with 9 data items per row.
The screen locations and field lengths shown are for the first row.
Screen locations for rows 2 through 15 are calculated by adding 100 to the screen location of the previous row for each of the data items.

```
10 REM ADDRESSING ROUTINE       CAMOBLDR.BAS
20 REM LATEST REVISIONS      01/16/91
30 DIM NAD(31)
40 OPEN "data.loc" AS #1 LEN=4:FIELD #1, 4 AS DL$
50 OPEN "data.fld" AS #2 LEN=4:FIELD #2, 4 AS DF$
60 OPEN "data.add" AS #3 LEN=128:FIELD #3, 128 AS DA$
70 COLOR 15,1:CLS
80 REM NUMBER OF DATA ITEMS PER SCREEN - NAD() ARRAY
90 DATA 11,27,9,9,9,9,9,9,9,9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
100 REM  SCREEN LOCATIONS FOR SITE SPECIFIC DATA (11 DATA ITEMS)
110 DATA 636,739,858,965,1354,1457,1539,1650,1744,1854,2154
120 REM SCREEN LOCATIONS FOR GEOMETRY DATA (27 DATA ITEMS)
130 DATA 648,927,932,937,942,947,952,957,962,967,972
160 DATA 1127,1132,1137,1142,1147,1152,1157,1162,1167,1172
180 DATA 1447,1549,1962,2048,2150,2260
200 REM FIRST SCREEN LOCATIONS FOR POLE MATERIALS DATA
210 DATA 902,904,911,920,929,938,947,956,965
220 REM SECOND SCREEN LOCATIONS FOR POLE MATERIALS DATA
230 DATA 902,904,911,920,929,938,947,956,965
240 REM FIRST SCREEN LOCATIONS FOR NETTING MATERIALS DATA
250 DATA 902,904,911,920,929,938,947,956,965
260 REM SECOND SCREEN LOCATIONS FOR NETTING MATERIALS DATA
270 DATA 902,904,911,920,929,938,947,956,965
280 REM FIRST SCREEN LOCATIONS FOR CABLING MATERIALS DATA
290 DATA 902,904,911,920,929,938,947,956,965
300 REM SECOND SCREEN LOCATIONS FOR CABLING MATERIALS DATA
310 DATA 902,904,911,920,929,938,947,956,965
320 REM FIRST SCREEN LOCATIONS FOR ANCHORS MATERIALS DATA
330 DATA 902,904,911,920,929,938,947,956,965
340 REM SECOND SCREEN LOCATIONS FOR ANCHORS MATERIALS DATA
350 DATA 902,904,911,920,929,938,947,956,965
360 REM FIELD LENGTHS FOR SITE SPECIFIC DATA (11 DATA ITEMS)
370 DATA 10,10,10,10,10,10,10,10,10,10,25
380 REM FIELD LENGTHS FOR GEOMETRY DATA (56 DATA ITEMS)
390 DATA 10,2,4,2,4,2,4,2,4,2,4,2,4,2,4,2,4,2,4,2,4,10,10,10,2,10,10
410 REM FIELD LENGTHS FOR SUPPORT MEMBERS DATA (FIRST SCREEN)
420 DATA 1,6,8,8,8,8,8,8,15
430 REM FIELD LENGTHS FOR POLE MATERIALS DATA (SECOND SCREEN)
440 DATA 1,6,8,8,8,8,8,8,15
450 REM FIELD LENGTHS FOR NETTING DATA (FIRST SCREEN)
460 DATA 1,6,8,8,8,8,8,8,15
470 REM FIELD LENGTHS FOR NETTING DATA (SECOND SCREEN)
480 DATA 1,6,8,8,8,8,8,8,15
490 REM FIELD LENGTHS FOR TENSION MEMBER DATA (FIRST SCREEN)
500 DATA 1,6,8,8,8,8,8,8,15
510 REM FIELD LENGTHS FOR TENSION MEMBER DATA (SECOND SCREEN)
520 DATA 1,6,8,8,8,8,8,8,15
530 REM FIELD LENGTHS FOR ANCHORS DATA (FIRST SCREEN)
540 DATA 1,6,8,8,8,8,8,8,15
550 REM FIELD LENGTHS FOR ANCHORS DATA (SECOND SCREEN)
```

```
560 DATA 1,6,8,8,8,8,8,8,15
570 REM
580 FOR I=1 TO 31:READ NAD(I):NEXT I
590 A$="":FOR I=1 TO 31:A$=A$+MKS$(NAD(I)):NEXT I
600 LSET DA$=A$:PUT #3,1:CLOSE #3
610 NR=0:FOR I=1 TO 31:IF NAD(I)=0 THEN 640
620 FOR J=1 TO NAD(I):NR=NR+1
630 READ A:LSET DL$=MKS$(A):PUT #1,NR:NEXT J
640 NEXT I:CLOSE #1
650 NR=0:FOR I=1 TO 31:IF NAD(I)=0 THEN 680
660 FOR J=1 TO NAD(I):NR=NR+1
670 READ A:LSET DF$=MKS$(A):PUT #2,NR:NEXT J
680 NEXT I:CLOSE #2:CLS
690 END
```

3 Utility Routines

The current version of the Camouflage Design Program uses 14 utility routines during the various program functions. The first 12 utility routines are stored in a single program module and are linked to each executable module involved with the program input, edit, and output functions. The two remaining routines are stand-alone executable routines which are used to create data files that the program execution requires. Each of these utility routines will be described and the source codes provided in this chapter. The utility routine names are as follows: (1) SCRNDUMP, (2) DISPLAY, (3) GETLOC, (4) GETFLD, (5) DSKWRT, (6) DSKRD, (7) CURSOR, (8) GENINP, (9) SCAN, (10) LOOKPRIC, (11) BLANK, (12) OUTDIR, (13) VALID, and (14) DEVICE.

SCRNDUMP Routine. The SCRNDUMP routine displays the specified interval of values from the VAR$() array to the screen. This routine is CALLed immediately after the screen is displayed, and fills the screen with the data items that are stored in the VAR$() array. Arguments to the routine are IST, the first value of the VAR$() array to be displayed, and IEND, the last value of the VAR$() array to be displayed. The individual data items are located on the screen from the corresponding values in the LPTS() array. The screen ROW number is determined from the first two characters of the LPTS() value (INT(LPTS(IZ)/100)), and the screen COLumn number is determined from the last two characters of the LPTS() value (INT(LPTS(IZ)-ROW*100)). The source codes for the routine are provided below:

```
 60 SUB SCRNDUMP(IST,IEND)
 70 FOR I=IST TO IEND:NSP=FLD(I)-LEN(VAR$(I)):IF NSP<=0 THEN 90
 80 FOR IZ=1 TO NSP:VAR$(I)=VAR$(I)+" ":NEXT IZ
 90 ROW=INT(LPTS(I)/100):COL=INT(LPTS(I)-ROW*100)
100 LOCATE ROW,COL:PRINT VAR$(I):NEXT I
110 END SUB
```

DISPLAY Routine. The DISPLAY routine is used to "highlight" a specified data field on the screen. This routine is used during the input or edit program functions to identify incorrect input when possible. The only argument to the routine is the number of the data item to be highlighted, IJ.

The screen location, ROW and COL, are determined from the corresponding value in the LPTS() array, and the specified data field is highlighted by using the COLOR function. The incorrect value of the data item is displayed on the screen after the field is highlighted so that the user may identify the source of the problem. The source codes of the routine are provided below:

```
120 SUB DISPLAY(IJ)
130 ROW=INT(LPTS(IJ)/100):COL=INT(LPTS(IJ)-ROW*100)
140 B$="":FOR J=1 TO FLD(IJ):B$=B$+" ":NEXT J:COLOR 31,0
150 LOCATE ROW,COL:PRINT B$:LOCATE ROW,COL:PRINT VAR$(IJ)
170 LOCATE ROW,COL:PRINT VAR$(IJ)
180 END SUB
```

GETLOC Routine. The GETLOC routine is used to extract the beginning screen field locations for the individual data items for a specified input screen. This routine is used immediately before an input screen is displayed. The arguments to the routine are the screen number, IP, and a single precision array, DUM(). The beginning screen field locations are stored in the random access data file DATA.LOC in sequential four-byte records. The DATA.LOC file must have been OPENed as device number 2 before the GETLOC routine is CALLed. The first record to be read (NST+1) is first determined by summing the number of data items (NAD()) associated with the input screens preceding input screen number IP. The number of records to be read is defined by NAD(IP). Each record is read, the beginning screen field location is extracted from the DL$ string with the CVS function, and the value is stored in the DUM() array. The source codes for the routine are provided below:

```
190 SUB GETLOC(IP,DUM(1))
200 NST=0:IF IP=1 THEN 220
210 FOR I=1 TO IP-1:NST=NST+NAD(I):NEXT I
220 IEN=NST+NAD(IP):NO=0:FOR I=NST+1 TO IEN:NO=NO+1
230 GET #2,I:DUM(NO)=CVS(DL$):NEXT I
240 END SUB
```

GETFLD Routine. The GETFLD routine is used to extract the field lengths for the individual data items for a specified input screen. This routine is used immediately before an input screen is displayed or when screen data are extracted from the project data file. The arguments to the routine are the screen number, IP, and a single precision array, DUM(). The field lengths are stored in the random access data file DATA.FLD in sequential four-byte records. The DATA.FLD file must have been OPENed as device number 3 before the GETFLD routine is CALLed. The first record to be read (NST+1) is first determined by summing the number of data items (NAD()) associated with the input screens preceding input screen number IP. The number of records to be read is defined by NAD(IP). Each record is read, the field length is extracted from the DF$ string with the CVS function, and the value is stored in the DUM() array. The source codes for the routine are provided below:

```
250 SUB GETFLD(IP,DUM(1))
260 NST=0:IF IP=1 THEN 280
270 FOR I=1 TO IP-1:NST=NST+NAD(I):NEXT I
280 IEN=NST+NAD(IP):NO=0:FOR I=NST+1 TO IEN:NO=NO+1
290 GET #3,I:DUM(NO)=CVS(DF$):NEXT I
300 END SUB
```

DSKWRT Routine. The DSKWRT routine is used to write the data stored in the VAR$() array to the project data file. This routine is used at the conclusion of each input screen during an input/edit program function. The only argument to the routine is the first 128-byte record number which is to be written. The project data file must be OPEN and designated as device number 1. In addition, the number of data items to be written (NLOC) and the field lengths of the data items (FLD()) must be defined. The number of data items that will be written is stored in the first four bytes of the first record to be written, and the number of bytes written (LGR) is set to four. Each data item is processed sequentially as follows:

(a) Check the length of the input data item (LEN(VAR$(I))); if the length is less than the defined field length, the VAR$() value is filled with blanks to the prescribed field length,

(b) Check the length (LGR) of the current string (RA$) to be written to ensure that a sufficient number of bytes exist to add the current data item,

(c) If sufficient space exists, add the current data item to the string and increase the string length by the field length of the data item, or

(d) If sufficient space does not exist, the current string is blank filled to 128 bytes and written to the project data file; the record count (FREC) is incremented by one; the current (I) data item is loaded into the first FLD(I) bytes of the record string and the string length set to FLD(I).

(e) When all of the data items have been processed, check the length of the current string (RA$); if the length of the string is zero, all the records have been written; if the length of the string is greater than zero, the string is blank filled to 128 bytes and written to the project data file.

The source codes for the routine are provided below:

```
310 SUB DSKWRT(FREC)
320 RA$=MKS$(NLOC):LGR=4
330 FOR I=1 TO NLOC:NA=FLD(I)-LEN(VAR$(I)):IF NA<=0 THEN 350
340 FOR J=1 TO NA:VAR$(I)=VAR$(I)+" ":NEXT J
350 IF LGR+FLD(I)>128 THEN 370
```

```
360 LGR=LGR+FLD(I):RA$=RA$+MID$(VAR$(I),1,FLD(I)):GOTO 410
370 LAD=128-LEN(RA$):IF LAD=0 THEN 390
380 FOR J=1 TO LAD:RA$=RA$+" ":NEXT J
390 LSET AA$=RA$:PUT #1,FREC:FREC=FREC+1
400 RA$=MID$(VAR$(I),1,FLD(I)):LGR=FLD(I)
410 NEXT I:IF LGR=0 THEN 430
420 LAD=128-LGR:FOR I=1 TO LAD:RA$=RA$+" ":NEXT I
425 LSET AA$=RA$:PUT #1,FREC
430 END SUB
```

DSKRD Routine. The DSKRD routine is used to retrieve the data stored in the project data file. This routine is used to retrieve the project data for a particular input screen. This data retrieval is accomplished during an edit program function to allow the data to be displayed before the edit process is begun or during an execution program function to define certain parameters for the design routines. The arguments to the routine are the first 128-byte records (FREC), the number of data items to be read (LOCN), the array of the field lengths of the data items (DUM()), and the array of the values of the data items (DUM$()). The project data file must be OPEN and designated as device number 1. In addition, the first record to be read (FREC) and the field lengths of the data items (DUM()) must be defined. The first record is read and loaded into a string RA$. The number of data items to be read, LOCN, is extracted from the first four bytes of that string using the CVS function. The number of bytes extracted from the string, LGR, is set to four. For each data item, a check is first made to ensure that sufficient space is left on the current string for the data item (I). If a sufficient number of bytes remain (LGR+DUM(I)<=128), the current data item is extracted from the input string RA$ from byte LGR+1 for DUM(I) bytes and loaded into the string array, DUM$(I). The number of bytes extracted (LGR) is incremented by DUM(I), and processing for the next data item is begun. If sufficient space does not exist in the current record for the data item (LGR+DUM(I)>128), the current number, FREC, is incremented by one, and the next record is read from the project data file and loaded into the string RA$. The number of types extracted, LGR, is set to zero. The data item is then extracted from the first DUM(I) bytes of the string RA$, LGR incremented by DUM(I), and processing for the next data item begun. The source codes for the routine are provided below:

```
440 SUB DSKRD (FREC,LOCN,DUM(1),DUM$(1))
450 LGR=4:GET #1,FREC:RA$=AA$:LOCN=CVS(MID$(RA$,1,4))
460 FOR I=1 TO LOCN:IF LGR+DUM(I)>128 THEN 480
470 DUM$(I)=MID$(RA$,LGR+1,DUM(I)):LGR=LGR+DUM(I):GOTO 490
480 FREC=FREC+1:GET #1,FREC:RA$=AA$:LGR=0:GOTO 470
490 NEXT I
500 END SUB
```

CURSOR Routine. The CURSOR routine is called by the GENINP routine to check each character of input provided by the user. These two routines are CALLed during the edit/input program functions for each input screen. The arguments to the CURSOR routine are the one-character input (B$), the data field number of the current cursor location (ILOC), and the data field number that the cursor is to be moved to (NXLOC). Eight possible cursor movement functions are defined in the INK() array. Each of these cursor movements is an extended two-character code representing a single keystroke. The first character is a NULL character which indicates an extended code character, and the second character is the actual keystroke character. The ASC function is used to extract the numeric equivalent of this second extended code character. The eight keystrokes that are accepted by the CURSOR routine and the ASCII extended code decimal equivalent of each are provided in Table 1.

Table 1
Extended Code Decimal Equivalents for Acceptable CURSOR Movements

Movement Number	Keyboard Description	ASCII Decimal Equivalent	NXLOC Value Returned
1	HOME	71	1
2	UP ARROW	72	ILOC-1
3	PAGE UP	73	1
4	LEFT ARROW	75	ILOC-1
5	RIGHT ARROW	77	ILOC+1
6	END	79	NLOC
7	DOWN ARROW	80	ILOC+1
8	PAGE DOWN	81	NLOC

Each user-specified keystroke (B$) is passed to the CURSOR routine to determine if the cursor should be moved to a different data field location. The ASC function is used to extract the ASCII decimal equivalent (NINK) of the second character (MID$(B$,2,1)) of B$. If one of the eight acceptable cursor movements as provided in Table 1 has been provided, the desired data field for cursor movement is loaded to NXLOC and returned to the GENINP routine. If an acceptable cursor movement is not identified, NXLOC is set to the current cursor location ILOC and returned to the GENINP routine.

```
510 SUB CURSOR(B$,ILOC,NXLOC)
520 DIM INK(8)
530 INK(1)=71:INK(2)=72:INK(3)=73:INK(4)=75
535 INK(5)=77:INK(6)=79:INK(7)=80
540 INK(8)=81:NINK=ASC(MID$(B$,2,1))
550 FOR I=1 TO 8:IF NINK=INK(I) THEN 570
```

```
560 NEXT I:NXLOC=ILOC:GOTO 730:REM NO FIND - IGNORE AND RETURN
570 ON I GOTO 580,590,580,640,660,680,690,680
580 NXLOC=1:GOTO 730:REM END SUB
590 IF ILOC=1 THEN NXLOC=1:GOTO 730
600 NXLOC=ILOC
610 NXLOC=NXLOC-1:IF NXLOC=1 THEN GOTO 730
620 IF INT(LPTS(ILOC)/100)=INT(LPTS(NXLOC)/100) THEN 610
630 GOTO 730
640 IF ILOC=1 THEN NXLOC=1:GOTO 730
650 NXLOC=ILOC-1:GOTO 730
660 IF ILOC=NLOC THEN NXLOC=NLOC:GOTO 730
670 NXLOC=ILOC+1:GOTO 730
680 NXLOC=NLOC:GOTO 730
690 IF ILOC=NLOC THEN NXLOC=NLOC:GOTO 730
700 NXLOC=ILOC
710 NXLOC=NXLOC+1:IF NXLOC=NLOC THEN GOTO 730
720 IF INT(LPTS(ILOC)/100)=INT(LPTS(NXLOC)/100) THEN 710
730 END SUB
```

GENINP Routine. The GENINP routine is the general input routine for the input/edit program functions. The routine is used to record each keystroke of input; check the input for extended code keystrokes and move the cursor to the designated data field, if appropriate; check the input for an <ESC> keystroke to terminate the input and move to the next screen; check the input for a <F1> keystroke to terminate the input and move to the previous screen; and load the input into the correct locations of the VAR$() array. The arguments to the routine are the first subscript of the VAR$() array to accept input (IST), the last subscript of the VAR$() array to accept input (IEND), and the ASCII decimal equivalent of the last keystroke made (LAST). Normal termination of the routine is indicated by the <ESC> key (LAST=27), and an abnormal termination of the routine is indicated by the <F1> key (LAST=59). Any other keystroke will be either accepted as input or ignored.

Each of the data fields from IST to IEND will be processed until either an <ESC> or <F1> keystroke is made. The variable ILOC is the current data field being processed. The variables ROW and COL contain the screen location for the beginning of the current data field. The string DFV$ contains the previously defined value of the current data item. Each keystroke is stored in the one-character string B$. The string A$ is used to concatenate the individual keystrokes for the data field. The variable COLC is the current column location of the cursor within the data field.

When a keystroke is provided, checks are made for an ASCII 27 (<ESC> key) or for an ASCII 8 (<BACKSPACE>). If <ESC> is detected, the routine is terminated and control returned to the CALLing routine. If <BACKSPACE> is detected, the length of the current input string (A$) and current cursor location (COLC) are checked to ensure that the cursor is not moved beyond the current data field limits. If the length of A$ is greater than zero, the right-most character is deleted and the column location COLC is decreased by one. The last keystroke LAST is set to 8.

If neither an <ESC> or <BACKSPACE> was detected, the next check is for an extended code (Table 1) keystroke. If the length of B$ is 2, an extended code keystroke has been made. If the extended code keystroke is ASCII 59 (<F1>), input is terminated and control is returned to the CALLing routine. If the extended code keystroke is less than ASCII 71 or greater than ASCII 81, an unacceptable extended code keystroke has been made, and the input is ignored. If an acceptable extended code keystroke is detected, the CURSOR routine is called to determine the location of the next data field, the current contents of the string A$ are stored in the ILOC position of the VAR$() array, and the cursor is moved to the next data field.

If an extended code keystroke was not detected, a check is made for an ASCII 13 (<ENTER>) keystroke. If an <ENTER> is detected, the current contents of the A$ string are stored in VAR$(ILOC), and the cursor is moved to the next (ILOC+1) data field. If an <ENTER> is not detected, normal data input is assumed. The keystroke B$ is added to the string A$, the cursor column location (COLC) is incremented by one, and the last keystroke (LAST) is set to the ASCII value of the B$ keystroke. This input process continues until an <ESC> or <F1> keystroke is made. The source codes for the routine are provided below:

```
740 SUB GENINP(IST,IEND,LAST)
750 ILOC=IST:LAST=27
760 ROW=INT(LPTS(ILOC)/100):COL=INT(LPTS(ILOC)-ROW*100)
765 DFV$=VAR$(ILOC)
770 LOCATE ROW,COL:COLOR 0,7
775 C$="":FOR I=1 TO FLD(ILOC):C$=C$+" ":NEXT I
780 PRINT C$:LOCATE ROW,COL:PRINT VAR$(ILOC):A$="":COLC=COL
790 LOCATE ROW,COLC,1:B$=INKEY$:IF LEN(B$)=0 THEN 790
800 IF ASC(B$)=27 THEN 1040:REM ESCAPE
810 IF ASC(B$)<>8 THEN  860:REM DESTRUCT BACKSPACE
820 IF LEN(A$)=0 THEN 790
830 A$=LEFT$(A$,LEN(A$)-1):LOCATE ROW,COL:PRINT C$
835 LOCATE ROW,COL:PRINT A$
840 IF COLC>COL THEN COLC=COLC-1
850 LAST=8:GOTO 790
860 IF LEN(B$)<>2 THEN 960:REM EXTENDED CODE KEYS HAVE LEN=2
870 IF ASC(MID$(B$,2,1))=59 THEN LAST=59:GOTO 1040
880 IF ASC(MID$(B$,2,1))<71 OR ASC(MID$(B$,2,1))>81 THEN 790
890 CALL CURSOR(B$,ILOC,NXLOC)
900 IF NXLOC=ILOC THEN 790
910 IF LEN(A$)<>0 THEN 940
920 LOCATE ROW,COL:COLOR 15,1:PRINT C$
930 LOCATE ROW,COL:PRINT DFV$:ILOC=NXLOC:GOTO 760
940 VAR$(ILOC)=A$:LOCATE ROW,COL
945 COLOR 15,1:PRINT C$:LOCATE ROW,COL
```

```
950 PRINT A$:ILOC=NXLOC:GOTO 760
960 IF ASC(B$)<>13 THEN 1010:REM CARRIAGE RETURN (ENTER KEY)
970 IF LEN(A$)=0 THEN A$=DFV$
980 VAR$(ILOC)=A$:LOCATE ROW,COL:COLOR 15,1:PRINT C$
990 LOCATE ROW,COL:PRINT A$:IF ILOC=IEND THEN 790
1000 ILOC=ILOC+1:GOTO 760
1010 IF LEN(A$)=>FLD(ILOC) THEN 1030
1020 A$=A$+B$:LOCATE ROW,COL:PRINT C$
1025 LOCATE ROW,COL:PRINT A$:COLC=COLC+1
1030 LAST=ASC(B$):GOTO 790
1040 IF LEN(A$)<>0 THEN 1070
1050 LOCATE ROW,COL:COLOR 15,1:PRINT C$
1060 LOCATE ROW,COL:PRINT DFV$:GOTO 1090
1070 VAR$(ILOC)=A$:LOCATE ROW,COL:COLOR 15,1:PRINT C$
1080 LOCATE ROW,COL:PRINT A$
1090 FOR I=IST TO IEND:NA=FLD(I)-LEN(VAR$(I)):IF NA<=0 THEN 1110
1100 FOR J=1 TO NA:VAR$(I)=VAR$(I)+" ":NEXT J
1110 NEXT I
1120 LOCATE ROW,COL,0:END SUB
```

SCAN Routine. The SCAN routine will extract the individual numeric and alpha-numeric data items from an input string (Epps and Corey 1990). This routine is CALLed during any of the program functions when numeric data must be extracted from the string arrays VAR$() or DUM$(). The arguments of the SCAN routine are an input string (A$), a single precision array (V()) for the returned numeric data, the number of numeric data items returned (NN), a string array (B$()) for the returned alpha-numeric data, and the number of alpha-numeric data items returned (NW).

The B$() and V() arrays and the NW and NN counters are initialized. Each character of the A$ string is examined for either a number (0-9), a plus (+), a minus (-), a decimal point (.) or a blank. The string variable VZ$ is used to store the contents of the number or word being processed. The first character of VZ$ is used to determine whether a number or word is being processed: if that first character is either a zero (0), a plus (+) sign, a minus (-) sign, a decimal point (.) or a numeric (0-9), a number is assumed and the variable NZ is set to zero. For any other characters, a word is assumed and NZ is set to two. A blank character signifies the end of the current number or word: the respective counter (NW or NN) is incremented by one, the contents of VZ$ are stored in the respective array (B$() or V()), and the input string VZ$ is blanked. This process continues until the end of the input string A$ is reached. The source codes for the routine are provided below:

```
1130 SUB SCAN(A$,V(1),NN,B$(1),NW)
1140 FOR IZ=1 TO 20:B$(IZ)="*":V(IZ)=0:NEXT IZ:VZ$="":A$=A$+" "
1150 NW=0:NN=0:NZ=0
1160 FOR IZ=1 TO LEN(A$)
1170 IZ$=MID$(A$,IZ,2):LZ$=LEFT$(IZ$,1):RZ$=RIGHT$(IZ$,1)
1180 IF NZ=2 THEN 1270
```

```
1190 IF NZ=1 THEN 1310
1200 IF LZ$=" " THEN 1340
1210 IF LZ$="0" THEN 1260
1220 IF LZ$="+" THEN 1260
1230 IF LZ$="-" THEN 1260
1240 IF LZ$="." THEN 1260
1250 IF LZ$<"1" OR LZ$>"9" GOTO 1290
1260 NZ=2
1270 VZ$=VZ$+LZ$:IF RZ$<>" " THEN 1340
1280 NN=NN+1:V(NN)=VAL(VZ$):VZ$="":NZ=0:GOTO 1340
1290 IF NZ<>0 THEN 1310
1300 B$(NW+1)=LZ$:NZ=1:GOTO 1320
1310 B$(NW+1)=B$(NW+1)+LZ$
1320 IF RZ$<>" " THEN 1340
1330 NW=NW+1:NZ=0
1340 NEXT IZ
1350 END SUB
```

LOOKPRIC Routine. The LOOKPRIC routine is used to search the pricing data file for the six-character identifier of a specified material item and return the desired pricing information for that material item. The routine is used in the edit/input program functions to search the pricing data file for material items for the support members, tension members, netting, and anchors. The arguments to the routine are the six-character material identifier (ITM$) and a string containing the material characteristics (PRST$) of the specified material item. The pricing data file must be OPEN as device number 5 before the LOOKPRIC routine can be called. In addition, the string array PRD$() must have been defined in the FIELD statement accompanying the OPEN statement. For a detailed discussion of the pricing data file organization, refer to Chapter 4 of this document.

The six-character material identifier (ITM$) consists of the five-character name of the material item and a one-character use code of the material item. The use code is used to categorize the individual material items as to support members (P), tension members (C), netting (N), anchors (A), or others. A complete list of the use codes currently defined is provided in Chapter 4. The sixth character of ITM$ is extracted as the use code US$, and the ASCII representation of US$ is stored in NOASC. The BEGAD() array in the unlabeled COMMON block contains the beginning subscript in the MARK() array of five-character material names for each use code. The variable NOASC is the subscript in the BEGAD() array for the use code specified. All material items with a blank code are listed first, followed by use codes 0-9 and use codes A-Z, providing a total of 37 use codes.

If no material items currently exist for the specified use code (BEGAD(NOASC)=0), execution of the routine is terminated, and a blank material string (PRST$) is returned to the CALLing routine. If material items exist for the specified use code, the MARK() array is searched over the subscript limits for the specified use code for the first five characters of ITM$. If the material name is found, the record number for the material item is determined, and the material data are extracted from the pricing data file and loaded into the string PRST$. If the material name is not found, a blank PRST$ string is returned. The source codes for the routine are provided below:

```
1840 SUB LOOKPRIC(ITM$,PRST$)
1850 REM
2100 PRST$=" "
2110 NOASC = 0                              ' 04/28/88
2120 NLEN = LEN(ITM$)                       '
2130 IF NLEN >= 6 THEN                      ' MODIFIED BY
     C.H. LIN
2140    US$=MID$(ITM$,6,1): NOASC=ASC(US$)  ' TO SKIP A
        NULL STRING
2150 END IF                                 '
2160 IF NOASC=32 THEN NOASC=1
2170 IF NOASC>=48 AND NOASC<=57 THEN NOASC=NOASC-46
2180 IF NOASC>=65 AND NOASC<=90 THEN NOASC=NOASC-53
2190 IF NOASC<=0 OR NOASC>37 THEN 2320
2200 IF BEGAD(NOASC)=0 THEN 2320
2210 NS=BEGAD(NOASC):IF NOASC=37 THEN NE=NMARK:GOTO 2260
2220 KT=NOASC
2230 NE=BEGAD(KT+1):IF NE>0 THEN 2260
2240 KT=KT+1:IF KT=37 THEN NE=NMARK:GOTO 2260
2250 GOTO 2230
2260 FOR IZ=NS TO NE:IF MID$(ITM$,1,5)=MID$(MARK(IZ),1,5)
     THEN 2280
2270 NEXT IZ:GOTO 2320
2280 NREC=IZ+79:GET #5,NREC
2290 REM
2300 PRST$=MID$(ITM$,1,6)+MID$(PRD$(2),1,4)
2310 REM
2320 END SUB
```

BLANK Routine. The BLANK routine is used to determine whether a specified data item is empty (blank) or not. This routine is used in all of the program functions. The arguments of the routine are the string value of the data field to be examined (SPEC$) and a blank indicator (NBLK). If the string is blank, NBLK is 0, and if the string is not blank, NBLK is 1. Each character of the SPEC$ string is examined for a blank (ASCII 32), and if a nonblank character is detected, NBLK is set to one and the execution of the routine is terminated. The source codes for the routine are shown below:

```
2330 SUB BLANK(SPEC$,NBLK)
2340 REM  PROGRAM TO CHECK BLANK  DATA RECORDS
2350 REM  SPEC$      STRING DATA ITEM TO BE CHECKED FOR BLANKS
2360 REM  NBLK       1    STRING DATA ITEM SPEC$ IS NOT BLANK
```

```
2370 NBLK=0:FOR JZ=1 TO LEN(SPEC$)
2380 IF ASC(MID$(SPEC$,JZ,1))<>32 THEN NBLK=1:GOTO 2400
2390 NEXT JZ
2400 END SUB
```

OUTDIR Routine. The OUTDIR routine is used to display a directory of the project data file which currently resides on the floppy disk drive. This routine is used at the beginning of an edit or execution program function when the program requires a project name and revision number be input for an existing project. If the user is not sure of the project name and/or the revision number, a response of DIR (or dir) will CALL the OUTDIR routine and display all of the project names and revision numbers of the project data files found on the floppy disk (A drive). The only argument of the routine is a string array (DUM$()), which contains the revision numbers of the project data files found on the floppy disk drive. The project names of the data files found on the floppy disk drive are returned in the VAR$() array, which resides in the unlabeled COMMON block. A total file directory of the floppy (A:) disk drive is output to a scratch file named DIRECT. Each file name (A$) in this scratch file is checked for a first character of P, the project data file name prefix character. If the first character of the file name is a P, the project name is extracted from the next six characters, the revision number extracted from the eighth character, and the data file extension is extracted from characters 10-12. If the data file extension is not .DAT, the file name is ignored and the directory search continued. If the extension name is .DAT, the file counter NODAT is incremented by one, the project name is stored in the VAR$() array, and the revision number is stored in the DUM$() array. When the file search is complete, the project names are sorted alphabetically, and the resulting list of project names and revision numbers are displayed on the screen. The source codes for this routine are provided below:

```
2850 SUB OUTDIR(DUM$(1))
2860 SHELL "DIR A: > DIRECT"
2870 CLS:OPEN "DIRECT" FOR INPUT AS #1
2880 LOCATE 1,25:PRINT "U. S. ARMY CORPS OF ENGINEERS"
2890 LOCATE 2,23:PRINT "DIRECTORY OF CAMOUFLAGE PROJECTS"
2900 PRINT
2910 FOR IZ=1 TO 3:PRINT "   PROJECT    REVISION    ";:NEXT IZ
2915 PRINT " "
2920 FOR IZ=1 TO 3:PRINT "    NUMBER     NUMBER     ";:NEXT IZ
2925 PRINT "    "
2930 NODAT=0
2940 IF EOF(1) THEN CLOSE #1:GOTO 3070
2950 LINE INPUT #1, A$
2960 IF MID$(A$,1,1)<>"P" THEN 2940
2970 IF LEN(A$)<12 THEN 2940
2980 ESN$=MID$(A$,2,6):RV$=MID$(A$,8,1):EXT$=MID$(A$,10,3)
2990 IF EXT$<>"DAT" THEN 2940
3000 IF NODAT=0 THEN 3030
```

```
3010 FOR IZ=1 TO NODAT:IF ESN$=VAR$(IZ) AND RV$=DUM$(IZ) THEN 2940
3020 NEXT IZ
3030 NODAT=NODAT+1:VAR$(NODAT)=ESN$:DUM$(NODAT)=RV$
3040 GOTO 2940
3050 REM
3060 REM  BUBBLE SORT PROJECT NAMES AND REVISION NUMBERS
3070 FOR IZ=1 TO NODAT-1:LOWP=VAL(VAR$(IZ))
3075 LOWR=VAL(DUM$(IZ)):ROW=IZ
3080 FOR JZ=IZ+1 TO NODAT:IF VAL(VAR$(JZ))>LOWP THEN 3120
3090 IF VAL(VAR$(JZ))<LOWP THEN 3110
3100 IF VAL(DUM$(JZ))>LOWR THEN 3120
3110 LOWP=VAL(VAR$(JZ)):LOWR=VAL(DUM$(JZ)):ROW=JZ
3120 NEXT JZ
3130 IF ROW=IZ THEN 3160
3140 PDUM$=VAR$(IZ):RDUM$=DUM$(IZ)
3145 VAR$(IZ)=VAR$(ROW):DUM$(IZ)=DUM$(ROW)
3150 VAR$(ROW)=PDUM$:DUM$(ROW)=RDUM$
3160 NEXT IZ
3170 FOR IZ=1 TO NODAT
3180 PRINT USING "    \   \         \\      ";VAR$(IZ),DUM$(IZ);
3190 IF IZ/3=INT(IZ/3) THEN PRINT " "
3200 NEXT IZ
3210 REM
3220 REM FINISHED - USER PROMPT TO CLEAR SCREEN
3230 LOCATE 24,2:PRINT "Press any key to continue ";
3240 A$=INKEY$:IF LEN(A$)=0 THEN 3240
3250 END SUB
```

VALID Program. The VALID program is a stand-alone executable program which is used to create the program data file COMFIL. The contents of the COMFIL data file are as follows:

(1) The maximum date for program validation; after this date has occurred, the program will automatically terminate and KILL the COMFIL data file, thereby preventing the program from being executed until the COMFIL data file is recreated;

(2) A 25-character user name;

(3) A five-character program registration number (1-99999) which will appear on each page of printed output;

(4) The default disk drive specification; the letter-designation for the disk drive on which the project data will be maintained during program execution and all program data files are assumed to reside; and

(5) The five-character program version number which will appear on each page of printed output.

The VALID program must be executed before the Camouflage Design Program is executed. When the five inputs described above are provided successfully, the COMFIL data file will be written on the hard disk, and execution of the Camouflage Design Program can be initiated. Each of the five data items is written as a 128-byte record in the random access data file. No further executions of the VALID program are required unless any of the five data items are to be changed. Subsequent executions of the program simply replace the contents of the COMFIL, and no back copies of this data file are retained. The source codes for the program are provided below:

```
 10 REM ROUTINE TO CREATE VALID  COMFIL  DATA FILE    VALID.BAS
 20 REM
 30 REM   CONTENTS OF  COMFIL
 40 REM   RANDOM ACCESS DATA FILE WITH FIELD LENGTH = 128 BYTES
 50 REM   RECORD 1     VALIDATION DATE - MONTH, DAY AND YEAR
 60 REM   RECORD 2     USER NAME (25 CHARS - MAY BE BLANK)
 70 REM   RECORD 3     PROGRAM REGISTRATION NUMBER (5 CHARS)
 80 REM   RECORD 4     SCRATCH DISK DRIVE SPECIFICATION (1 CHAR)
 90 REM   RECORD 5     PROGRAM VERSION NUMBER (5 CHARS)
100 REM
110 DIM LPTS(5),FLD(5),VAR$(5)
120 DIM DATS(10),JUL(12)
130 COMMON SHARED /VALID/NLOC,FLD(),LPTS(),AA$,VAR$()
140 DATA 936,1136,1336,1536,1736,10,25,5,1,5
150 DATA 0,31,59,90,120,151,181,212,243,273,304,334
160 FOR I=1 TO 10:READ DATS(I):NEXT I
170 FOR I=1 TO 12:READ JUL(I):NEXT I
180 FOR I=1 TO 5:LPTS(I)=DATS(I):FLD(I)=DATS(I+5):NEXT I
190 REM
200 FOR IZ=1 TO 5:VAR$(IZ)="":FOR JZ=1 TO FLD(IZ)
210 VAR$(IZ)=VAR$(IZ)+" ":NEXT JZ:NEXT IZ
220 OPEN "COMFIL" AS #1 LEN=128:FIELD #1, 128 AS BA$
230 GET #1,1:VERN=CVS(BA$):YR=INT(VERN/1000):IF VERN=0 THEN 360
240 YRS=STR$(YR)
250 IF MID$(YR$,1,1)=" " THEN YR$=RIGHT$(YR$,LEN(YR$)-1):GOTO 250
260 DAT=VERN-YR*1000:FOR IZ=1 TO 12:IF JUL(IZ)>=DAT THEN 280
270 NEXT IZ:IZ=13
280 MON=IZ-1:DAY=DAT-JUL(MON)
290 MON$=STR$(MON):IF MON<10 THEN MID$(MON$,1,1)="0":GOTO 310
300 IF MID$(MON$,1,1)=" " THEN
302      MON$=RIGHT$(MON$,LEN(MON$)-1)
304      GOTO 300
306 ENDIF
310 DAY$=STR$(DAY):IF DAY<10 THEN MID$(DAY$,1,1)="0":GOTO 330
320 IF MID$(DAY$,1,1)=" " THEN
322      DAY$=RIGHT$(DAY$,LEN(DAY$)-1)
324      GOTO 320
326 ENDIF
330 VAR$(1)=MON$+"/"+DAY$+"/"+YR$
```

```
340 GET #1,2:VAR$(2)=MID$(BA$,1,25):GET #1,3:VAR$(3)
    =STR$(CVS(BA$))
350 GET #1,4:VAR$(4)=MID$(BA$,1,1):GET #1,5:VAR$(5)=MID$(BA$,1,5)
360 CLOSE #1
370 COLOR 15,1,1:CLS
380 PRINT:PRINT "U. S. Army Corps of Engineers"
390 PRINT "Waterways Experiment Station"
400 PRINT "Vicksburg, MS 39180"
410 PRINT:PRINT "Camouflage Design Program Validation"
420 LOCATE 9,12:PRINT "PROGRAM RENEWAL DATE"
430 LOCATE 11,23:PRINT "USER NAME"
440 LOCATE 13,5:PRINT "PROGRAM REGISTRATION NUMBER"
450 LOCATE 15,14:PRINT "SCRATCH DISK DRIVE"
460 LOCATE 17,10:PRINT "PROGRAM VERSION NUMBER"
470 LOCATE 24,2:PRINT "Press  <ESC>  when input is
    complete . . .";
480 IST=1:IEND=5:NLOC=IEND:CALL SCRNDUMP(IST,IEND)
490 CALL GENINP(IST,IEND,LAST)
500 REM
510 REM CHECK INPUT DATA FOR CORRECT FORMAT
520 DAT$=VAR$(1):NSL=0:FOR IZ=1 TO LEN(DAT$)
530 IF MID$(DAT$,IZ,1)="/" THEN NSL=NSL+1
540 NEXT IZ:IF NSL=2 THEN 630
550 REM
560 REM RENEWAL DATE HAS INCORRECT FORMAT
570 B$=" Renewal Date must have following format:  MO/DA/YEAR "
580 COLOR 31,1,1:LOCATE 24,2:PRINT B$;
590 A$=INKEY$:IF LEN(A$)=0 THEN 590
600 B$="                                                        "
610 COLOR 15,1,1:LOCATE 24,2:PRINT B$;:GOTO 470
620 REM
630 MON=VAL(MID$(DAT$,1,2)):IF MON<1 OR MON>12 THEN 570
640 DAY=VAL(MID$(DAT$,4,2)):IF DAY<1 OR DAY>31 THEN 570
650 YEAR=VAL(MID$(DAT$,7,4)):IF YEAR<1988 THEN 570
660 REM
670 REM GOOD DATE HAS BEEN PROVIDED
680 REM PROGRAM REG NUMBER MUST BE PROVIDED
690 DNUM=VAL(VAR$(3)):IF DNUM>=1 AND DNUM<=99999! THEN 730
700 B$=" License Number must be specified - Press any key
    to continue"
710 GOTO 580
720 REM
730 IF LAST<>27 THEN 480
740 OPEN "COMFIL" AS #1 LEN=128:FIELD #1, 128 AS BA$
750 VERN=YEAR*1000+JUL(MON)+DAY
760 LSET BA$=MKS$(VERN):PUT #1,1
770 LSET BA$=MID$(VAR$(2),1,25):PUT #1,2
780 LSET BA$=MKS$(DNUM):PUT #1,3
790 LSET BA$=MID$(VAR$(4),1,1):PUT #1,4
800 LSET BA$=MID$(VAR$(5),1,5):PUT #1,5
```

```
810 CLOSE #1:CLS
820 END
```

DEVICE Program. The DEVICE program is a stand-alone executable program that is used to create the *DEVICES* data file. *The DEVICES* data file is a sequential file which contains two records (lines) when properly loaded. The data from this file are used to define the type of display and the type of printer being used by the Camouflage program. The contents of the DEVICES data file are as follows:

(1) The keyword PRINT followed by the port number and model number for the printer type specified and scaling factor that will be used for the printer plots generated by the program; and

(2) The keyword SCREEN followed by the port number and model number for the display type specified and the scaling factor that will be used for the graphic displays produced by the program.

The DEVICE program must be executed before the Camouflage Design Program is executed. When the required inputs are provided successfully, the DEVICES data file will be written on the hard disk, and execution of the Camouflage Design Program can be initiated. No further executions of the DEVICE program are required unless the printer or the display on the computer is changed. Subsequent executions of the program simply replace the contents of the DEVICES, and no back copies of this data file are retained. An example of the contents of the DEVICES data file and the source codes for the DEVICE program are provided below:

```
PRINT    0 11  1.0000   12  17
SCREEN  91 91  1.0000   11
```

```
 10 REM CAMOUFLAGE DESIGN DEVICES SPECIFICATION      DEVICE.BAS
 20 REM   LATEST REVISION  01/29/91 JWE
 30 DIM LPTS(135),FLD(135),VAR$(135),IREC(31),NAD(31),PRD$(18)
 40 DIM BEGAD(37),MARK(2002) AS STRING * 5
 50 COMMON SHARED NLOC,NAD(),FLD(),LPTS(),IREC(),NMARK,
         BEGAD(),DL$,DF$,AA$,VAR$(),PRD$(),MARK() AS STRING*5
 55 COLOR 15,1,1:CLS
 60 ON ERROR GOTO 1160
 70 DIM DLOC(3),DUMR$(18),V(20),B$(20)
 80 DATA 536,538,576
 90 NPRT=0:FOR IZ=1 TO 3:READ DLOC(IZ):NEXT IZ
100 NR=1:NRZ=17:NM=1:MD=0
110 REM
120 OPEN "DEVICES" FOR INPUT AS #1
130 IF EOF(1) THEN 180
140 LINE INPUT #1, A$:CALL SCAN(A$,V(),NN,B$(),NW)
150 IF MID$(B$(1),1,4)="PRIN" THEN NR=V(4):NRZ=V(5)
160 IF MID$(B$(1),1,4)="SCRE" THEN NM=V(4)
170 GOTO 130
```

```
180 CLOSE #1
190 REM
200 REM MONITOR/DISPLAY SPECIFICATION
210 NLOC=12:LPTS(1)=DLOC(1):FLD(1)=1
220 FOR IZ=2 TO NLOC:LPTS(IZ)=LPTS(IZ-1)+100:FLD(IZ)=1:NEXT IZ
230 FOR IZ=1 TO NLOC:VAR$(IZ)="N":NEXT IZ:VAR$(1)="Y"
240 IF NM<>0 THEN VAR$(1)="N":VAR$(NM)="Y"
250 IST=1:IEND=NLOC:CALL MONITOR:CALL SCRNDUMP(IST,IEND)
260 LOCATE 24,2:PRINT "Press  <ESC>  when input is
    complete . . . ";
270 CALL GENINP(IST,IEND,LAST)
280 REM
290 REM DATA CHECKING FOR CONSISTENCY
300 REM CHECK FOR ONLY ONE MONITOR SPECIFICATION
310 IYES=0:FOR IZ=1 TO NLOC
320 IF VAR$(IZ)="Y" OR VAR$(IZ)="y" THEN IYES=IYES+1:NM=IZ
330 NEXT IZ:IF IYES=1 THEN 400
340 B$(1)=" Only ONE   MONITOR DEVICE   must be specified -
    Press any key  "
350 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 350
360 COLOR 31,1,1:LOCATE 24,2:PRINT B$(1);
370 A$=INKEY$:IF LEN(A$)=0 THEN 370
380 COLOR 15,1,1:GOTO 250
390 REM
400 IF LAST<>27 THEN 270
410 CALL DISPLAY(NM):COLOR 15,1,1
420 A$=INKEY$:IF LEN(A$)=0 THEN 420
430 REM
440 REM MONITOR DISPLAY DEVICE NUMBERS = PORT NUMBERS
450 MODM=0:SCFCT=1!:IF NM=2 THEN MODM=93
460 IF NM=3 OR NM=5 THEN MODM=99
470 IF NM=4 OR NM=8 THEN MODM=94
480 IF NM=6 THEN MODM=95
490 IF NM=7 THEN MODM=96
500 IF NM=9 THEN MODM=97
510 IF NM=10 THEN MODM=90
520 IF NM=11 THEN MODM=91
530 IF NM=12 THEN MODM=92
540 REM
550 REM PRINTER/DISPLAY SPECIFICATIONS
560 NLOC=18:LPTS(1)=DLOC(1)
565 FOR IZ=2 TO 16:LPTS(IZ)=LPTS(IZ-1)+100:NEXT IZ
570 LPTS(17)=DLOC(3):LPTS(18)=LPTS(17)+100
580 FOR IZ=1 TO NLOC:FLD(IZ)=1:NEXT IZ
590 FOR IZ=1 TO NLOC:VAR$(IZ)="N":NEXT IZ
600 VAR$(NR)="Y":VAR$(NRZ)="Y"
610 IF NPRT=0 THEN 660
620 REM
630 REM LOAD LAST DISPLAY SPECIFICATIONS
640 FOR IZ=1 TO NLOC:VAR$(IZ)=DUMR$(IZ):NEXT IZ
```

```
650 REM
660 IST=1:IEND=NLOC:CALL PRINTER:CALL SHEETS
665 CALL SCRNDUMP(IST,IEND)
670 LOCATE 24,2
680 PRINT "Press  <ESC>  to complete input or  <F1>  for previous
    screen ...";
690 CALL GENINP(IST,IEND,LAST)
700 NPRT=1:FOR IZ=1 TO NLOC:DUMR$(IZ)=VAR$(IZ):NEXT IZ
710 IF LAST=59 THEN 210
720 REM
730 REM DATA CHECKING FOR CONSISTENCY
740 REM CHECK  FOR ONLY ONE PRINTER SPECIFICATION
750 IYES=0:FOR IZ=1 TO 16
760 IF VAR$(IZ)="Y" OR VAR$(IZ)="y" THEN IYES=IYES+1:NR=IZ
770 NEXT IZ:IF IYES=1 THEN 850
780 B$(1)=" Only  ONE  PRINTER DEVICE   must be specified -
    Press any key  "
790 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 790
800 COLOR 31,1,1:LOCATE 24,2:PRINT B$(1);
810 A$=INKEY$:IF LEN(A$)=0 THEN 810
820 COLOR 15,1,1:GOTO 660
830 REM
840 REM CHECK  FOR ONLY ONE PRINTER PAGE SIZE SPECIFICATION
850 IYES=0:FOR IZ=17 TO 18
860 IF VAR$(IZ)="Y" OR VAR$(IZ)="y" THEN IYES=IYES+1:NRZ=IZ
870 NEXT IZ:IF IYES=1 THEN 910
880 B$(1)=" Only  ONE  Printer PAGE SIZE  must be specified
    - Press any key  "
890 GOTO 790
900 REM
910 IF LAST<>27 THEN 560
920 CALL DISPLAY(NR):CALL DISPLAY(NRZ):COLOR 15,1,1
930 A$=INKEY$:IF LEN(A$)=0 THEN 930
940 REM
950 REM PRINTER DEVICE NUMBERS - ALL PORT NUMBERS = 0
960 MODR=1:IF NR=3 OR NR=4 THEN MODR=5
970 IF NR=7 OR NR=12 THEN MODR=11
980 IF NR=5 OR NR=6 THEN MODR=15
990 IF NR=8 THEN MODR=41
1000 IF NR=16 THEN MODR=60
1010 IF NR=13 THEN MODR=70
1020 IF NR=14 THEN MODR=72
1030 IF NR=15 THEN MODR=75
1040 REM
1050 REM CALCULATE PAGE SIZE FACTOR FROM SPECIFIED PAPER SIZE
1060 PRFCT=1!:IF NRZ=18 THEN PRFCT=14!/11!
1070 REM
1090 REM WRITE IPORT  IDEV   FACT  TO FILE   'DEVICES'  ON
     HARD DISK
1100 CLS:OPEN "DEVICES" FOR OUTPUT AS #1:PRPRT=0
```

```
1110 A$="PRINT #### ## ##.####  ### ###"
1115 PRINT #1, USING A$;PRPRT,MODR,PRFCT,NR,NRZ
1120 PRINT #1, USING "SCREEN ### ## ##.####  ###";
     MODM,MODM,SCFCT,NM
1130 CLOSE:GOTO 1260
1140 REM
1150 REM ERROR CHECKING
1160 IF ERR<>53 THEN 1210
1170 OPEN "DEVICES" FOR OUTPUT AS #1:PRINT #1, " ":CLOSE #1
1180 RESUME 210
1190 REM
1200 REM GENERAL ERROR - ABORT PROGRAM
1210 B$(1)=" Error Number #### - at Line Number ##### - Press
     any key"
1220 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 1220
1230 COLOR 31,1,1:LOCATE 24,2:PRINT USING B$(1);ERR,ERL;
1240 A$=INKEY$:IF LEN(A$)=0 THEN 1240
1250 COLOR 15,1,1:CLS
1260 END
1270 REM
1280 REM PRINTER/DISPLAY DEVICES SPECIFICATION SCREEN
1290 SUB PRINTER
1300 CLS
1310 LOCATE 1,25:PRINT "U. S. ARMY CORPS OF ENGINEERS"
1320 LOCATE 2,23:PRINT "CAMOUFLAGE DESIGN OUTPUT DEVICES"
1330 LOCATE 4,3:PRINT "PRINTING DEVICE                (Y/N)"
1340 LOCATE 5,5:PRINT "NO PRINTER"
1350 LOCATE 6,5:PRINT "EPSON (MX,RX-80)"
1360 LOCATE 7,5:PRINT "EPSON (FX,JX-80)"
1370 LOCATE 8,5:PRINT "EPSON (FX-85)"
1380 LOCATE 9,5:PRINT "EPSON (FX-185,286)"
1390 LOCATE 10,5:PRINT "EPSON (FX-100)"
1400 LOCATE 11,5:PRINT "EPSON (MX,RX-100)"
1410 LOCATE 12,5:PRINT "EPSON (LQ-1500)"
1420 LOCATE 13,5:PRINT "IBM (PRPRNTR/GRPHCS)"
1430 LOCATE 14,5:PRINT "CENTRONICS"
1440 LOCATE 15,5:PRINT "OKIDATA (92,182,192)"
1450 LOCATE 16,5:PRINT "OKIDATA (93,193)"
1460 LOCATE 17,5:PRINT "HP THINKJET"
1470 LOCATE 18,5:PRINT "HP QUIETJET"
1480 LOCATE 19,5:PRINT "HP QUIETJET PLUS"
1490 LOCATE 20,5:PRINT "HP LASERJET"
1500 END SUB
1510 SUB MONITOR
1520 CLS
1530 LOCATE 1,25:PRINT "U. S. ARMY CORPS OF ENGINEERS"
1540 LOCATE 2,23:PRINT "CAMOUFLAGE DESIGN OUTPUT DEVICES"
1550 LOCATE 4,3:PRINT "DISPLAY ADAPTER                (Y/N)"
1560 LOCATE 5,5:PRINT "TEXT MONITOR ONLY"
1570 LOCATE 6,5:PRINT "HERCULES MONO (720 BY 348)"
```

```
1580 LOCATE 7,5:PRINT "CGA ( 2 COLOR 640 BY 200)"
1590 LOCATE 8,5:PRINT "EGA (16 COLOR 320 BY 200)"
1600 LOCATE 9,5:PRINT "EGA ( 2 COLOR 640 BY 200)"
1610 LOCATE 10,5:PRINT "EGA (16 COLOR 640 BY 200)"
1620 LOCATE 11,5:PRINT "EGA (MONOCHRM 640 BY 350)"
1630 LOCATE 12,5:PRINT "EGA (16 COLOR 320 BY 200)"
1640 LOCATE 13,5:PRINT "EGA (16 COLOR 640 BY 350)"
1650 LOCATE 14,5:PRINT "VGA (2 COLOR 640 BY 480)"
1660 LOCATE 15,5:PRINT "VGA (16 COLOR 640 BY 480)"
1670 LOCATE 16,5:PRINT "VGA (256 COLOR 320 BY 200)"
1680 END SUB
1690 SUB SHEETS
1700 LOCATE 4,51:PRINT "PAGE SIZE            (Y/N)"
1710 LOCATE 5,54:PRINT "8.5 BY 11.0  (A)"
1720 LOCATE 6,53:PRINT "11.0 BY 14.0 (AA)"
1730 LOCATE 7,53:PRINT "11.0 BY 17.0  (B)"
1740 LOCATE 8,53:PRINT "17.0 BY 22.0  (C)"
1750 LOCATE 9,53:PRINT "22.0 BY 34.0  (D)"
1760 END SUB
```

Reference.

Epps, James W., and Corey, Marion W. (1990 (Jul)). "Computerized applications for station equations," *Journal of Computing in Civil Engineering*, ASCE, 4(3),269-278.

4 Pricing Data

The pricing data file for the Camouflage Design program is named PRICE.LAT and contains the structural and costing characteristics of each item of material that the program uses. The purpose of this chapter is to provide a detailed description of how this data file is organized, how the data for individual material items within the data file are extracted by the program, and how additional material data items may be incorporated into the pricing data file. The pricing data file is not necessary for program execution, but its existence will greatly enhance the input program function when the primary project material's structural and costing characteristics must be provided.

Pricing Data File Organization. The pricing data for the program are contained in a random access data file named PRICE.LAT, which contains 130-byte records. The data file is divided into three major sections, as follows:

(1) The first two 130-byte records contain 38 four-byte single precision values. The first 37 values are the beginning record numbers, which contain the five-character names of the material items contained in the pricing file for each of the 37 acceptable use codes. The 38th value is the total number of material items contained in the pricing data file. With 130-byte records, the first 32 values of the beginning record numbers are contained in the first record, leaving two blank bytes at the end of the record. The second 130-byte record contains the remaining five beginning record numbers and the total number of material items contained in the pricing data file, leaving 106 blank bytes at the end of that record. The data from these two records are maintained in the BEGAD() array and the NMARK variable. Both of these variables are contained in the unlabeled COMMON block. At the beginning of each major program module, these data must be extracted from the PRICE.LAT data file using the CVS function and stored in these variables so that the data are available to the other routines within the program module. The source codes for this initial data extraction (from the MAIN program) are provided below:

```
620 OPEN DD$+"PRICE.LAT" AS #5 LEN=130:FIELD #5, 130 AS BG$
630 GET #5,1:FOR I=1 TO 32:JZ=(I-1)*4+1
640 BEGAD(I)=CVS(MID$(BG$,JZ,4)):NEXT I
650 GET #5,2:FOR I=33 TO 37:JZ=(I-33)*4+1
660 BEGAD(I)=CVS(MID$(BG$,JZ,4)):NEXT I:NMARK=CVS(MID$(BG$,21,4))
```

(2) The next 77 records (record #3 through record #79) contain the five-character names of the individual material items whose data are contained in the pricing data file. A maximum of 2,002 material items are allowed, requiring a total of 10,100 bytes. At 130 bytes per record, a total of 77 records are required. These five-character names are extracted from record numbers 3 through 79 and stored in the string array MARK, which is contained in the unlabeled COMMON block. The source codes for this data extraction (from the MAIN program) are provided below:

```
670 FOR I=3 TO 79:GET #5,I:IJ=(I-3)*26:FOR JZ=1 TO 26
680 KZ=(JZ-1)*5+1:MARK(IJ+JZ)=MID$(BG$,KZ,5):NEXT JZ:NEXT I
```

(3) The remaining records (record #80 through a maximum of record #2081) contain the structural, costing, and descriptive data for each material item in the pricing data file. These data are not loaded into any program arrays like the previous data were, but are accessed directly from the data file as they are needed. The organization of each of these 130-byte records must be defined before any of the material characteristics can be extracted from the data file, however. Each 130-byte record contains 18 data items associated with each material. The number of bytes associated with each of these data items is stored in the LNG() array and defined with a READ/DATA statement at the beginning of each major program module. The LNG() is contained in the GENERAL-labeled COMMON block. The source codes for the initialization of the LNG() array and the description of the file record organization are provided below:

```
260 DATA 6,4,4,2,4,2,4,30,30,4,4,4,4,4,4,8,4,8
280 FOR I=1 TO 18:READ LNG(I):NEXT I

700 FOR MZ=1 TO 18:SM=0:IF MZ=1 THEN 720
710 FOR NZ=1 TO MZ-1:SM=SM+LNG(NZ):NEXT NZ
720 FIELD #5, SM AS DUMM$, LNG(MZ) AS PRD$(MZ):NEXT MZ
```

The relative location within the 130-byte record, description, and field length of each of the 18 individual material characteristics are provided below:

Data Item	Description	Field Length
1	ITEM MARK	6
2	PRIMARY CATEGORY	4
3	MATERIAL SIZE (LENGTH OR AREA)	4
4	PRICING UNIT OF MEASURE	2
5	ALLOWABLE STRENGTH (YIELD)	4
6	ESTIMATE UNIT OR MEASURE	2
7	ESTIMATE UNIT WEIGHT (WET OR DRY)	4
8	FEDERAL STOCK NUMBER	30
9	FABRICATED PART/USE DESCRIPTION	30
10	MODULUS OF ELASTICITY	4
11	CROSS-SECTIONAL AREA OR THICKNESS	4
12	MOMENT OF INERTIA	4
13	COEFFICIENT OF THERMAL EXPANSION	4
14	ANCHOR SIZE (HELIX, INCHES)	4
15	ITEM UNIT PRICE	4
16	ERECTION TIME (MAN-HRS)	8
17	DRAG COEFFICIENT (NET ONLY)	4
18	CURRENTLY NOT IN USE	8

The MARK of each data item is made up of the five-character name of the material data item and the one-character use code. The MARK name can be any five characters, but the use code must correspond to the assigned category of the data item. The acceptable use codes are as follows: (1) a blank character, (2) the numerics 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9, and (3) the alpha-numerics A through Z. The total of 37 acceptable use codes correspond to the maximum subscript on the BEGAD() array which is located in the unlabeled COMMON block. Care must be taken in the assignment of the use code for a material item because the material screens for support members, tension members, netting, and anchors are initially filled during the input program function from the pricing data file (Chapter 5 CAMINP Routine). A complete description of the material use codes is provided in a later section of this chapter which describes the **EDITPRIC** program. This program allows user to add material data items to the existing version of the PRICE.LAT data file. The use codes provided in Table 2 have been assigned to the current version of the PRICE.LAT data file:

Table 2
Current Pricing Data File Use Codes and Descriptions

Use Code	Description
A	Anchors and related materials
B	Bolts and primary connectors
C	Structural tension members (cables)
E	Eyebolts
M	Pulleys
N	Netting
P	Structural support members
R	Rings and snap connectors
T	Turnbuckles

The PRIMARY CATEGORY of each item of material is a four-character descriptor which is used to further categorize each data item. The only use of this data item is in the material summaries which are displayed in the output routines. The material SIZE is a four-byte numeric for the unit size of the item. The PRICING UNIT OF MEASURE is a two-character descriptor which indicates how the item is priced, i.e., by the foot, square yard, square foot, pound, etc. The ALLOWABLE STRENGTH of the item is a four-byte numeric which represents the yield or tearing strength of the item expressed in kips/square inch (ksi). The ESTIMATE UNIT OF MEASURE is a two-character descriptor which indicates how the item is priced in the Camouflage Program. The pricing and estimate units of measure should be the same for each data item. The UNIT WEIGHT of the item is the four-byte numeric weight per unit of measure for the data item, i.e., pounds/foot, pounds/square yard, etc. The FEDERAL STOCK NUMBER and FABRICATED PART DESCRIPTION are 30-character descriptors of the item. No current use is being made of the federal stock number, but later versions of the program can adjust inventory levels if desired. The part description is displayed both in the input screens and the final program output. The MODULUS OF ELASTICITY is a four-byte numeric in kips/square inch for the structural items and anchors only. The CROSS-SECTION AREA OR THICKNESS is a four-byte numeric in square inches for the structural items or inches for the netting items. The MOMENT OF INERTIA is a four-byte numeric in inches^4 for the support members only. The COEFFICIENT OF THERMAL EXPANSION is a four-byte numeric in inches (x 10^6) per degree F for the tension members only. The ANCHOR SIZE is a four-byte numeric for the diameter of the helix, in inches, for the anchors only. The UNIT PRICE is the four-byte unit cost of the item in dollars per unit of measure. The ERECTION TIME is a four-byte numeric for the unit erection time in man-hours for the support members and anchors only. The erection time for the support members should include the excavation time for the hole and establishment of the base support system. This data item is not being used in the current version of the program, however. The DRAG COEFFICIENT is the four-byte numeric for the netting materials only. The last data item is not defined in the current version of the program, but provides an eight-byte space at the end of each 130-byte record.

Material Data Extraction. The individual data items associated with a single material item are extracted from the pricing data file, PRICE.LAT, using the LOOKPRIC routine as described in Chapter 3 Utility Routines of this document. Before any such data extraction can occur, however, the following program functions must be accomplished:

(1) The PRICE.LAT data file must have been OPENed as a random access file with 130-byte records with an accompanying FIELD statement,

(2) The BEGAD() (single precision beginning record numbers for each use code) and MARK() (five-character material names) and the variable NMARK (single precision total number of material items) must have been loaded from the PRICE.LAT data file,

(3) The LNG() array must have been defined either by DATA/READ statements or by arithmetic assignment statements, and

(4) The PRD$() array must have been defined by the FIELD statement for the individual material item records (record numbers 80 through 2081).

The source codes required to accomplish the four program functions are provided below:

```
80 DIM DUM(135),DUM$(135),V(20),B$(20),LNG(18)
90 COMMON SHARED/GENERAL/DUM(),V(),LNG(),DUM$(),B$(),A$
260 DATA 6,4,4,2,4,2,4,30,30,4,4,4,4,4,4,8,4,8
280 FOR I=1 TO 18:READ LNG(I):NEXT I

620 OPEN DD$+"PRICE.LAT" AS #5 LEN=130:FIELD #5, 130 AS BG$
630 GET #5,1:FOR I=1 TO 32:JZ=(I-1)*4+1
640 BEGAD(I)=CVS(MID$(BG$,JZ,4)):NEXT I
650 GET #5,2:FOR I=33 TO 37:JZ=(I-33)*4+1
660 BEGAD(I)=CVS(MID$(BG$,JZ,4)):NEXT I:NMARK=CVS(MID$(BG$,21,4))
670 FOR I=3 TO 79:GET #5,I:IJ=(I-3)*26:FOR JZ=1 TO 26
680 KZ=(JZ-1)*5+1:MARK(IJ+JZ)=MID$(BG$,KZ,5):NEXT JZ:NEXT I
690 REM
700 FOR MZ=1 TO 18:SM=0:IF MZ=1 THEN 720
710 FOR NZ=1 TO MZ-1:SM=SM+LNG(NZ):NEXT NZ
720 FIELD #5, SM AS DUMM$, LNG(MZ) AS PRD$(MZ):NEXT MZ
```

When these four program functions have been accomplished, the PRICE.LAT data file may be searched for any six-character (five-character name and one-character use code) material name using the LOOKPRIC routine. The two arguments to the LOOKPRIC routine are the six-character name to locate and the string variable PRST$ which contain the six-character material name and four-character primary category of the material item. If the name of the material item is not found in the PRICE.LAT data file, a blank string PRST$ is returned. The 18 individual material characteristics are stored in the PRD$() array, which is contained in the unlabeled COMMON block, and may be extracted into other data arrays for use by the program as desired. The source codes which illustrate the extraction of these data during the input program function are provided below:

```
2100 DIM POLNAME$(30)
2110 NPOL=0:NS=BEGAD(27):IF NS=0 THEN 2170
2120 NE=BEGAD(28)-1
2130 FOR I=NS TO NE:GET #5,I+79
2140 NPOL=NPOL+1:POLNAME$(NPOL)=PRD$(1)
2150 NEXT I
2160 REM
2170 KEYZ=0:NPR=3:NYES=0
2270 NSTRT=1:IF NPR=4 THEN NSTRT=16
2280 NEND=NSTRT+14
2290 FOR I=NSTRT TO NEND:IJ=(I-NSTRT)*NAD(NPR)
2300 FOR J=1 TO NAD(NPR):VAR$(IJ+J)=" ":NEXT J
2310 IF I>NPOL THEN 2380
2320 VAR$(IJ+1)="N"
2330 VAR$(IJ+2)=POLNAME$(I):CALL LOOKPRIC(POLNAME$(I),PRST$)
2340 VAR$(IJ+3)=STR$(CVS(PRD$(3))):VAR$(IJ+4)=STR$(CVS(PRD$(5)))
2350 VAR$(IJ+5)=STR$(CVS(PRD$(10))):VAR$(IJ+6)=STR$(CVS(PRD$(11)))
2360 VAR$(IJ+7)=STR$(CVS(PRD$(12))):VAR$(IJ+8)=STR$(CVS(PRD$(15)))
2370 VAR$(IJ+9)=MID$(PRD$(9),1,15)
2380 NEXT I:GOTO 2420
2390 REM
2400 CALL DSKRD(FREC,NDUM,DUM(),DUM$())
2410 REM
2420 CALL POLSCRN(PN$,RN$):IST=1:IEND=NLOC
```

The source codes shown provide for the extraction of the material data contained in the PRICE.LAT data file relating to the support members. The variable NPOL contains the number of support members extracted from the pricing data file. The use code for the support members is 'P', and the beginning address for the material items with this use code (NS) is stored in BEGAD(27). Note that the blank use code, 10 numeric use codes, and use codes A through O occupy the first 26 addresses of the BEGAD() array. The ending record number (NE) for use code 'P' is one less than the beginning record nu.nber for use code 'Q' stored in BEGAD(28). For each record contained between NS and NE, the record is extracted using the GET #5 function, and the six-character name is extracted from PRD$(1) and stored in the program array POLNAME$(). Up to 30 six-character names may be extracted in this manner. In later sections of the source codes, the material characteristics of each name in the POLNAME$() array are extracted using the LOOKPRIC routine. For each item found in the pricing data file, the critical characteristics of each support member are extracted from the PRD$() array and stored in the VAR$() array. This will allow the data to be retained during subsequent CALLs to the LOOKPRIC routine, and also eventually will allow the data to be stored in the project data file when the DSKWRT routine is CALLed.

Addition of Material Items to the Pricing Data File. In the first section of this chapter, the statement was made that the contents of the PRICE.LAT data file were not essential for the proper execution of the current version of the Camouflage Program. If the PRICE.LAT data file was nonexistent (or empty), the user would be required to input at least one support member, one tension member, one netting material, and one anchor with each material's designated characteristics. Once these characteristics are provided during an input program function, they are retained only for the project name and revision number specified, and are not available for other project names and/or revision numbers. Only material data items that are currently stored in the PRICE.LAT data file are available for all project names and revision numbers. This section is intended to describe how one or more material data items can be included in the PRICE.LAT data file.

The stand-alone executable program **EDITPRIC** has been provided in the library of program modules, and is to be used to add material names and associated data to the contents of the pricing data file. The following conventions must be adhered to while adding data to the pricing data file:

(1) Existing material MARK names should not be duplicated; if the MARK name exists in the current version of the PRICE.LAT data file, subsequent data inputs will replace the current data in the data file. The first five characters of the MARK name can be any combination of letters and/or numbers or symbols, but the user is encouraged to use a systematic naming convention to avoid later confusion. The sixth character of the MARK name will be the use code of the data item.

(2) The use code for the material item to be added must follow the convention of the current list of use codes incorporated in the program (see Table 2 above). If a new use code is to be defined, a record of this use code must be maintained, so that when similar materials (with the same use code) are added, the same use code will be used. Simply stated: if an additional support member is to be included in the PRICE.LAT data file, the use code for that item must be specified as 'P'. The same applies for the other items listed in Table 2 above.

(3) Careful attention must be given to the required units of each data item. The current version of the Camouflage Program

assumes that the convention unit of distance is provided in feet, area in square feet, material strengths in either kips or kips per square inch, and so on.

The user's guide for the **EDITPRIC** program is provided in Appendix A. The source codes for the EDITPRIC program are provided below:

```
10 REM    PRICE DATA FILE MAINTENANCE PROGRAM    EDITPRIC.BAS
20 REM    LATEST REVISION     01/14/91 JWE
30 REM
40 REM          ITEM                                FIELD LENGTH
50 REM    1.  ITEM MARK                                   6
60 REM    2.  PRIMARY CATEGORY                            4
70 REM    3.  MATERIAL SIZE (LENGTH OR AREA)              4
80 REM    4.  PRICING UNIT OF MEASURE                     2
90 REM    5.  ALLOWABLE STRENGTH (YIELD)                  4
100 REM   6.  ESTIMATE UNIT OF MEASURE                    2
110 REM   7.  ESTIMATE UNIT WEIGHT (WET OR DRY)           4
120 REM   8.  FEDERAL STOCK NUMBER                       30
130 REM   9.  FABRICATED PART/USE DESCRIPTION            30
140 REM  10.  MODULUS OF ELASTICITY                       4
150 REM  11.  CROSS-SECTION AREA OR THICKNESS             4
160 REM  12.  MOMENT OF INERTIA                           4
170 REM  13.  COEFFICIENT OF THERMAL EXPANSION            4
180 REM  14.  ANCHOR SIZE (HELIX, INCHES)                 4
190 REM  15.  ITEM UNIT PRICE                             4
200 REM  16.  ERECTION TIME (MAN-HRS)                     8
210 REM  17.  DRAG COEFFICIENT (NET ONLY)                 4
220 REM  18.  NOT IN USE CURRENTLY                        8
230 REM
240 REM              TOTAL BYTES PER ITEM               130
250 REM
260 REM    THE DATA ARE STORED BY THE PRC$() ARRAY - DIMED
    PRC$(18)
270 REM
280 REM PRICE.LAT FILE ORGANIZATION - TOTAL OF 2081 RECORDS
290 REM   RECORD 1 - BEGAD(1) THRU BEGAD(32) - 4 BYTES EACH (128)
300 REM   RECORD 2 - BEGAD(33) THRU BEGAD(37) & NMARK -  24 BYTES
310 REM   RECORDS 3 THRU 79 - FIVE CHARACTER MARKS AT 26/RECORD
320 REM         NOTE:  26*5=130    26*77=2002
330 REM   RECORDS 80 THRU 2081 (2002) - MATERIAL DATA FOR EACH
    MARK
340 REM         NOTE:  ONE MARK PER RECORD (130 BYTES)
350 REM
360 DIM LPTS(135),FLD(135),VAR$(135),IREC(31),NAD(31),PRD$(18)
370 DIM BEGAD(37),MARK(2002) AS STRING*5
380 COMMON SHARED NLOC,NAD(),FLD(),LPTS(),IREC(),
      NMARK,BEGAD(),DL$,DF$,AA$,VAR$(),PRD$(),MARK() AS STRING*5
390 COLOR 15,1,1:CLS
400 LOCATE 5,25:PRINT "PRICING DATA GENERATION"
```

```
410 LOCATE 7,35:PRINT "for"
420 LOCATE 9,23:PRINT "U. S. Army Corps of Engineers"
430 LOCATE 14,29:PRINT "P. O. Box 631"
440 LOCATE 16,26:PRINT "Vicksburg, MS 39180"
450 LOCATE 19,29:PRINT "Copyright 1990"
460 LOCATE 24,2:PRINT "Press any key to continue . . .";
470 A$=INKEY$:IF LEN(A$)=0 THEN 470
480 REM
490 REM
500 DIM LNG(18),V(20),B$(20),DUM(135),DUM$(135),
505 DIM LOCS(18),ASCI$(37),NOITM(37)
510 DATA 6,4,4,2,4,2,4,30,30,4,4,4,4,4,4,8,4,8
520 DATA 546,646,746,846,946,1046,1146,1246,1346,1446,1546,
    1646,1746,1846
530 DATA 1946,2046,2146,2246
540 DATA " ","0","1","2","3","4","5","6","7","8","9","A","B","C",
    "D","E","F"
550 DATA "G","H","I","J","K","L","M","N","O","P","Q","R","S","T",
    "U","V","W"
560 DATA "X","Y","Z"
570 FOR IZ=1 TO 18:READ LNG(IZ):NEXT IZ
580 FOR IZ=1 TO 18:READ LOCS(IZ):NEXT IZ
590 FOR IZ=1 TO 37:READ ASCI$(IZ):NEXT IZ
600 OPEN "COMFIL" AS #1 LEN=128:FIELD #1, 128 AS BA$
610 GET #1,1:RNDT=CVS(BA$):GET #1,3:DLNUM=CVS(BA$)
620 GET #1,4:DD$=MID$(BA$,1,1)+":":GET #1,5:PGVER$=BA$
630 CLOSE #1
640 REM
650 REM INITIALIZE PRICING VARIABLES
660 FOR IZ=1 TO 37:BEGAD(IZ)=0:NOITM(IZ)=0:NEXT IZ:NMARK=0
670 FOR IZ=1 TO 2002:MARK(IZ)="     ":NEXT IZ
680 REM
690 NLOC=18:FOR I=1 TO NLOC:LPTS(I)=LOCS(I):NEXT I
700 FOR I=1 TO NLOC:FLD(I)=LNG(I):NEXT I
710 OPEN DD$+"PRICE.LAT" AS #1 LEN=130:FIELD #1, 130 AS BG$
720 GET #1,1:FOR IZ=1 TO 32:JZ=(IZ-1)*4+1
730 BEGAD(IZ)=CVS(MID$(BG$,JZ,4)):NEXT IZ
740 GET #1,2:FOR IZ=33 TO 37:JZ=(IZ-33)*4+1
750 BEGAD(IZ)=CVS(MID$(BG$,JZ,4)):NEXT IZ:NREC=CVS(MID$(BG$,21,4))
760 IF NREC=0 THEN 850
770 FOR IZ=3 TO 79:GET #1,IZ:IJ=(IZ-3)*26:FOR JZ=1 TO 26
780 KZ=(JZ-1)*5+1:MARK(IJ+JZ)=MID$(BG$,KZ,5):NEXT JZ:NEXT IZ
790 FOR IZ=1 TO 36:IF BEGAD(IZ)=0 THEN 810
800 NOITM(IZ)=BEGAD(IZ+1)-BEGAD(IZ)
810 NEXT IZ
820 IF BEGAD(37)>0 THEN NOITM(37)=NREC-BEGAD(37)+1
830 REM
840 REM
850 CLS:A$="INPUT, EDIT, DELETE, DISPLAY OR END (IN/ED/DE/DI/END)"
855 PRINT:PRINT A$;:INPUT A$
```

```
860 IF MID$(A$,1,2)="EN" OR MID$(A$,1,2)="en" THEN 3640
870 IF MID$(A$,1,2)="ED" OR MID$(A$,1,2)="ed" THEN 1880
880 IF MID$(A$,1,2)="DE" OR MID$(A$,1,2)="de" THEN 2560
890 IF MID$(A$,1,2)="DI" OR MID$(A$,1,2)="di" THEN 3300
900 IF MID$(A$,1,2)="IN" OR MID$(A$,1,2)="in" THEN 970
910 COLOR 31,1,1:LOCATE 24,2
920 PRINT "Input must be   IN/ED/DE/END  -  Try again  ";
930 A$=INKEY$:IF LEN(A$)=0 THEN 930
940 COLOR 15,1,1:GOTO 850
950 REM
960 REM INPUT PROGRAM FUNCTION
970 IF DLNUM=1 THEN 1020
980 B$=" INput Function not allowed at this station - Use
    EDit only"
990 COLOR 31,1,1:LOCATE 24,2:PRINT B$;
1000 A$=INKEY$:IF LEN(A$)=0 THEN 1000
1010 COLOR 15,1,1:GOTO 850
1020 FOR IZ=1 TO 18:IZS=0:IF IZ=1 THEN 1040
1030 FOR JZ=1 TO IZ-1:IZS=IZS+LNG(JZ):NEXT JZ
1040 FIELD #1, IZS AS DUMM$, LNG(IZ) AS PRD$(IZ):NEXT IZ
1050 REM
1060 CLS:PRINT:INPUT "Input  Item Mark Index or END";A$
1070 IF A$="END" OR A$="end" THEN 1750
1080 CALL BLANK(A$,NBLK):IF NBLK=0 THEN 1060
1090 IF LEN(A$)=6 THEN 1110
1100 JZ=6-LEN(A$):FOR IZ=1 TO JZ:A$=A$+" ":NEXT IZ
1110 US$=MID$(A$,6,1):NOASC=ASC(US$)
1120 IF NOASC=32 THEN NOASC=1
1130 IF NOASC>=48 AND NOASC<=57 THEN NOASC=NOASC-46
1140 IF NOASC>=65 AND NOASC<=90 THEN NOASC=NOASC-53
1150 IF NOASC>=97 AND NOASC<=122 THEN NOASC=NOASC-85
1160 IF NOASC<=0 THEN 1210
1170 IF BEGAD(NOASC)=0 THEN BEGAD(NOASC)=1
1180 NS=BEGAD(NOASC):GOTO 1270
1190 REM
1200 REM INVALID MARK INPUT
1210 B$=" Item Mark Index   \    \   is invalid - TRY AGAIN"
1220 COLOR 31,1,1
1230 LOCATE 24,2:PRINT USING B$;A$;
1240 A$=INKEY$:IF LEN(A$)=0 THEN 1240
1250 COLOR 15,1,1:GOTO 1060
1260 REM
1270 IF NOITM(NOASC)=0 THEN IZ=NS:GOTO 1380
1280 NE=NS+NOITM(NOASC)-1
1290 FOR IZ=NS TO NE:IF MID$(A$,1,5)<MARK(IZ) THEN 1380
1300 NEXT IZ:IZ=NE+1
1310 REM
1320 REM   IZ IS RECORD NUMBER FOR NEW RECORD
1330 REM   ALL DATA RECORDS FROM IZ TO NREC MUST BE MOVED DOWN
1335 REM       ONE (+79)
```

```
1340 REM   EACH MARK() FROM IZ TO NREC MUST BE MOVED DOWN ONE
1350 REM   EACH BEGAD() FROM NOASC+1 TO 37 MUST BE INCREASED BY
1355 REM  ONE
1360 REM   NREC MUST BE INCREASED BY ONE
1370 REM
1380 IF NOASC=37 THEN 1440
1390 REM ADJUST BEGAD() ARRAY
1400 FOR KZ=NOASC+1 TO 37
1405 IF BEGAD(KZ)> 0 THEN BEGAD(KZ)=BEGAD(KZ)+1
1410 IF BEGAD(KZ)=0 THEN BEGAD(KZ)=2
1420 NEXT KZ
1430 REM SHIFT RECORDS FROM IZ TO NREC DOWN ONE RECORD
1440 IF NREC=0 THEN 1480
1450 FOR KZ=NREC TO IZ STEP -1:MARK(KZ+1)=MARK(KZ)
1460 KREC=KZ+79:GET #1,KREC:PUT #1,KREC+1:NEXT KZ
1470 REM INCREASE NREC
1480 NREC=NREC+1:NOITM(NOASC)=NOITM(NOASC)+1:NOREC=IZ+79
1490 FOR KZ=1 TO 18:VAR$(KZ)=" ":NEXT KZ:VAR$(1)=MID$(A$,1,6)
1500 IST=1:IEND=NLOC
1505 CALL PRICING(NOREC):CALL SCRNDUMP(IST,IEND)
1510 LOCATE 24,2:PRINT "Press <ESC> when input is complete . . .";
1520 CALL GENINP(IST,IEND,LAST):IF LAST< > 27 THEN 1520
1530 REM
1540 REM CHECK TO INSURE MARK HAS NOT BEEN CHANGED
1550 IF VAR$(1)=MID$(A$,1,6) THEN 1630
1560 COLOR 31,1,1:LOCATE 24,2
1570 B$(1)=" Mark  \    \  cannot be changed - Use EDit Function"
1580 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 1580
1590 PRINT USING B$(1);MID$(A$,1,6);
1600 C$=INKEY$:IF LEN(C$)=0 THEN 1600
1610 VAR$(1)=MID$(A$,1,6):COLOR 15,1,1:GOTO 1500
1620 REM
1630 LSET PRD$(1)=VAR$(1):LSET PRD$(2)=VAR$(2)
1635 LSET PRD$(3)=MKS$(VAL(VAR$(3)))
1640 LSET PRD$(4)=VAR$(4):LSET PRD$(5)=MKS$(VAL(VAR$(5)))
1645 LSET PRD$(6)=VAR$(6)
1650 LSET PRD$(7)=MKS$(VAL(VAR$(7)))
1660 LSET PRD$(8)=VAR$(8):LSET PRD$(9)=VAR$(9)
1670 FOR KZ=10 TO 15:LSET PRD$(KZ)=MKS$(VAL(VAR$(KZ))):NEXT KZ
1680 LSET PRD$(16)=VAR$(16):LSET PRD$(17)=MKS$(VAL(VAR$(17)))
1690 LSET PRD$(18)=VAR$(18):PUT #1,NOREC
1700 MARK(IZ)=MID$(VAR$(1),1,5)
1710 REM
1720 GOTO 1060
1730 REM
1740 REM END OF INPUT - UPDATE BEGAD() ARRAY IN RECORDS 1 AND 2
1750 FIELD #1, 130 AS BG$
1760 A$="":FOR IZ=1 TO 32:A$=A$+MKS$(BEGAD(IZ)):NEXT IZ
1770 LSET BG$=A$:PUT #1,1
1780 A$="":FOR IZ=33 TO 37:A$=A$+MKS$(BEGAD(IZ)):NEXT IZ
```

```
1790 A$=A$+MKS$(NREC):LSET BG$=A$:PUT #1,2
1800 REM
1810 REM  UPDATE MARK() IN RECORDS 3 THRU 79
1820 FOR IZ=3 TO 79:IJ=(IZ-3)*26+1:IK=IJ+25
1830 A$="":FOR JZ=IJ TO IK:A$=A$+MARK(JZ):NEXT JZ
1840 LSET BG$=A$:PUT #1,IZ:NEXT IZ
1850 GOTO 850
1860 REM EDIT PROGRAM FUNCTION
1870 REM
1880 IF NREC > 0 THEN 1960
1890 COLOR 31,1,1:LOCATE 24,2
1900 B$(1)=" PRICE.LAT data file does not exist - Use
     INput Function"
1910 IF LEN(B$(1)) <78 THEN B$(1)=B$(1)+" ":GOTO 1910
1920 PRINT B$(1);
1930 C$=INKEY$:IF LEN(C$)=0 THEN 1930
1940 COLOR 15,1,1:GOTO 850
1950 REM
1960 FOR IZ=1 TO NLOC:IZS=0:IF IZ=1 THEN 1980
1970 FOR JZ=1 TO IZ-1:IZS=IZS+LNG(JZ):NEXT JZ
1980 FIELD #1, IZS AS DUMM$, LNG(IZ) AS PRD$(IZ):NEXT IZ
1990 CLS:PRINT:INPUT "Input  Item Mark Index or END";A$
2000 IF A$="END" OR A$="end" THEN 850
2010 CALL BLANK(A$,NBLK):IF NBLK=0 THEN 1990
2020 IF LEN(A$)=6 THEN 2040
2030 JZ=6-LEN(A$):FOR IZ=1 TO JZ:A$=A$+" ":NEXT IZ
2040 US$=MID$(A$,6,1):NOASC=ASC(US$)
2050 IF NOASC=32 THEN NOASC=1
2060 IF NOASC>=48 AND NOASC<=57 THEN NOASC=NOASC-46
2070 IF NOASC>=65 AND NOASC<=90 THEN NOASC=NOASC-53
2080 IF NOASC>=97 AND NOASC<=122 THEN NOASC=NOASC-85
2090 IF NOASC<=0 THEN 2160
2100 IF BEGAD(NOASC)=0 THEN 2160
2110 NS=BEGAD(NOASC):IF NOITM(NOASC)=0 THEN 2160
2120 NE=NS+NOITM(NOASC)-1
2130 FOR IZ=NS TO NE:IF MID$(A$,1,5)=MARK(IZ) THEN 2240
2140 NEXT IZ
2150 REM DID NOT FIND PRM$ IN CURRENT PRICING LIST
2160 B$=" Item Mark Index   \    \   not in pricing data
     - TRY AGAIN"
2170 COLOR 31,1,1
2180 LOCATE 24,2:PRINT USING B$;A$;
2190 A$=INKEY$:IF LEN(A$)=0 THEN 2190
2200 COLOR 15,1,1:GOTO 1990
2210 REM
2220 REM ITEM MARK INDEX FOUND IN PRICE.LAT FILE - DATA IN RECORD
2225 REM     NUMBER IZ+79
2230 REM LOAD RECORD INTO VAR$() ARRAY AND DISPLAY FOR EDIT
2240 NOREC=IZ+79:GET #1,NOREC
2250 FOR IZ=1 TO 2:VAR$(IZ)=PRD$(IZ):NEXT IZ
```

```
2260 VAR$(3)=STR$(CVS(PRD$(3))):VAR$(4)=PRD$(4)
2270 VAR$(5)=STR$(CVS(PRD$(5))):VAR$(6)=PRD$(6)
2280 VAR$(7)=STR$(CVS(PRD$(7)))
2290 FOR IZ=8 TO 9:VAR$(IZ)=PRD$(IZ):NEXT IZ
2300 FOR IZ=10 TO 15:VAR$(IZ)=STR$(CVS(PRD$(IZ))):NEXT IZ
2310 VAR$(16)=PRD$(16):VAR$(17)=STR$(CVS(PRD$(17)))
2320 VAR$(18)=PRD$(18)
2330 IST=1:IEND=NLOC
2335 CALL PRICING(NOREC):CALL SCRNDUMP(IST,IEND)
2340 LOCATE 24,2:PRINT "Press  <ESC>  when input is
     complete . . .";
2350 CALL GENINP(IST,IEND,LAST):IF LAST< >27 THEN 2350
2360 REM
2370 REM CHECK TO INSURE MARK HAS NOT BEEN CHANGED
2380 IF VAR$(1)=MID$(A$,1,6) THEN 2460
2390 COLOR 31,1,1:LOCATE 24,2
2400 B$(1)=" Mark  \    \   cannot be changed - Use
     INput Function"
2410 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 2410
2420 PRINT USING B$(1);MID$(A$,1,6);
2430 C$=INKEY$:IF LEN(C$)=0 THEN 2430
2440 VAR$(1)=MID$(A$,1,6):COLOR 15,1,1:GOTO 2330
2450 REM
2460 LSET PRD$(1)=VAR$(1):LSET PRD$(2)=VAR$(2)
2465 LSET PRD$(3)=MKS$(VAL(VAR$(3)))
2470 LSET PRD$(4)=VAR$(4):LSET PRD$(5)=MKS$(VAL(VAR$(5)))
2475 LSET PRD$(6)=VAR$(6)
2480 LSET PRD$(7)=MKS$(VAL(VAR$(7)))
2485 LSET PRD$(17)=MKS$(VAL(VAR$(17)))
2490 LSET PRD$(8)=VAR$(8):LSET PRD$(9)=VAR$(9):LSET
     PRD$(16)=VAR$(16)
2500 FOR IZ=10 TO 15:LSET PRD$(IZ)=MKS$(VAL(VAR$(IZ))):NEXT IZ
2510 LSET PRD$(16)=VAR$(16):LSET PRD$(17)=MKS$(VAL(VAR$(17)))
2520 LSET PRD$(18)=VAR$(18):PUT #1,NOREC
2530 GOTO 1990
2540 REM
2550 REM DELETE PROGRAM FUNCTION
2560 IF NREC>0 THEN 2640
2570 COLOR 31,1,1:LOCATE 24,2
2580 B$(1)=" PRICE.LAT data file does not exist - Use
     INput Function"
2590 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 2590
2600 PRINT B$(1);
2610 C$=INKEY$:IF LEN(C$)=0 THEN 2610
2620 COLOR 15,1,1:GOTO 850
2630 REM
2640 IF DLNUM=1 THEN 2670
2650 B$=" DElete Function not allowed at this station - Use
     EDit only"
2660 GOTO 990
```

```
2670 FOR IZ=1 TO NLOC:IZS=0:IF IZ=1 THEN 2690
2680 FOR JZ=1 TO IZ-1:IZS=IZS+LNG(JZ):NEXT JZ
2690 FIELD #1, IZS AS DUMM$, LNG(IZ) AS PRD$(IZ):NEXT IZ
2700 CLS:PRINT:INPUT "Input  Item Mark Index or END";A$
2710 IF A$="END" OR A$="end" THEN 1750
2720 CALL BLANK(A$,NBLK):IF NBLK=0 THEN 2700
2730 IF LEN(A$)=6 THEN 2750
2740 JZ=6-LEN(A$):FOR IZ=1 TO JZ:A$=A$+" ":NEXT IZ
2750 US$=MID$(A$,6,1):NOASC=ASC(US$)
2760 IF NOASC=32 THEN NOASC=1
2770 IF NOASC>=48 AND NOASC<=57 THEN NOASC=NOASC-46
2780 IF NOASC>=65 AND NOASC<=90 THEN NOASC=NOASC-53
2790 IF NOASC>=97 AND NOASC<=122 THEN NOASC=NOASC-85
2800 IF NOASC<=0 THEN 2870
2810 IF BEGAD(NOASC)=0 THEN 2870
2820 NS=BEGAD(NOASC):IF NOITM(NOASC)=0 THEN 2870
2830 NE=NS+NOITM(NOASC)-1
2840 FOR IZ=NS TO NE:IF MID$(A$,1,5)=MARK(IZ) THEN 2950
2850 NEXT IZ
2860 REM DID NOT FIND PRM$ IN CURRENT PRICING LIST
2870 B$=" Item Mark Index   \     \   not in pricing data - TRY
     AGAIN"
2880 COLOR 31,1,1
2890 LOCATE 24,2:PRINT USING B$;A$;
2900 A$=INKEY$:IF LEN(A$)=0 THEN 2900
2910 COLOR 15,1,1:GOTO 2700
2920 REM
2930 REM  ITEM MARK INDEX FOUND IN PRICE.LAT FILE - DATA
     IN  IZ+79
2935 REM        RECORD
2940 REM  LOAD CONTENTS OF RECORD INTO VAR$() ARRAY AND DISPLAY
2945 REM        FOR EDIT
2950 NOREC=IZ+79:GET #1,NOREC
2960 FOR IZ=1 TO 2:VAR$(IZ)=PRD$(IZ):NEXT IZ
2970 VAR$(3)=STR$(CVS(PRD$(3))):VAR$(4)=PRD$(4)
2980 VAR$(5)=STR$(CVS(PRD$(5))):VAR$(6)=PRD$(6)
2990 VAR$(7)=STR$(CVS(PRD$(7)))
3000 FOR IZ=8 TO 9:VAR$(IZ)=PRD$(IZ):NEXT IZ
3010 FOR IZ=10 TO 15:VAR$(IZ)=STR$(CVS(PRD$(IZ))):NEXT IZ
3020 VAR$(16)=PRD$(16):VAR$(17)=STR$(CVS(PRD$(17)))
3030 VAR$(18)=PRD$(18)
3040 IST=1:IEND=NLOC
3045 CALL PRICING(NOREC):CALL SCRNDUMP(IST,IEND)
3050 LOCATE 24,2:PRINT "Press  <ESC>  when input is
     complete . . .";
3060 CALL GENINP(IST,IEND,LAST):IF LAST< >27 THEN 3060
3070 REM
3080 REM CHECK TO INSURE MARK HAS NOT BEEN CHANGED
3090 IF VAR$(1)=MID$(A$,1,6) THEN 3170
3100 COLOR 31,1,1:LOCATE 24,2
```

```
3110 B$(1)=" Mark  \     \   cannot be changed - Use INput
     Function"
3120 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 3120
3130 PRINT USING B$(1);MID$(A$,1,6);
3140 C$=INKEY$:IF LEN(C$)=0 THEN 3140
3150 VAR$(1)=MID$(A$,1,6):COLOR 15,1,1:GOTO 3040
3160 REM
3170 B$(1)=" Delete this record:  Mark  \     \   -  Are you
     sure?? "
3180 LOCATE 24,2:PRINT USING B$(1);A$;:LINE INPUT C$
3190 IF MID$(C$,1,1)="Y" OR MID$(C$,1,1)="y" THEN 3210
3200 GOTO 2700
3210 IF NOASC=37 THEN 3230
3220 FOR KZ=NOASC+1 TO 37:BEGAD(KZ)=BEGAD(KZ)-1:NEXT KZ
3230 IF IZ=NREC THEN 3270
3240 FOR KZ=IZ TO NREC-1:MARK(KZ)=MARK(KZ+1):NEXT KZ
3250 FOR KZ=IZ TO NREC-1:KREC=KZ+79
3260 GET #1,KREC+1:PUT #1,KREC:NEXT KZ
3270 NREC=NREC-1:NOITM(NOASC)=NOITM(NOASC)-1:GOTO 2700
3280 REM
3290 REM DISPLAY PROGRAM OPTION
3300 CLS:PRINT SPC(31);:PRINT "PRICE.LAT  CONTENTS"
3310 FOR IZ=1 TO 3:PRINT SPC(2);
3315 PRINT "NUM  CHAR  BEGAD  NOITEM";
3320 NEXT IZ:PRINT " "
3330 A$=" ##  \\      ####   ####"
3340 FOR IZ=1 TO 12:FOR JZ=1 TO 3
3345 PRINT SPC(2);:ISUB=(JZ-1)*12+IZ
3350 PRINT USING A$;ISUB,ASCI$(ISUB),BEGAD(ISUB),NOITM(ISUB);
3360 NEXT JZ:PRINT " ":NEXT IZ
3370 ISUB=37:PRINT SPC(54);
3380 PRINT USING A$;ISUB,ASCI$(ISUB),BEGAD(ISUB),NOITM(ISUB)
3390 PRINT:PRINT USING "   NO. RECORDS   #### ";NREC
3400 LOCATE 24,2
3410 A$=INKEY$:IF LEN(A$)=0 THEN 3410
3420 REM
3430 CLS:PRINT:INPUT "LIST MARKS FOR USE CODE   OR END ";A$
3440 IF A$="END" OR A$="end" THEN 850
3450 CALL BLANK(A$,NBLK):IF NBLK=0 THEN IZ=1:GOTO 3490
3460 FOR IZ=2 TO 37:IF MID$(A$,1,1)=ASCI$(IZ) THEN 3490
3470 NEXT IZ:GOTO 3430
3480 REM
3490 IF NOITM(IZ)>0 THEN 3530
3500 PRINT:PRINT "   NO MARKS FOR USE CODE  ";ASCI$(IZ)
3510 GOTO 3590
3520 REM
3530 PRINT:FOR JZ=1 TO NOITM(IZ):ISUB=BEGAD(IZ)+JZ-1
3540 PRINT USING "    ####   \    \  ";ISUB,MARK(ISUB)
3550 IF INT(JZ/20)=JZ/20 THEN
3560     LOCATE 24,2:PRINT "  Press any key to continue ";
```

```
3570     A$=INKEY$:IF LEN(A$)=0 THEN 3570
3580     CLS
3590 ENDIF
3600 NEXT JZ
3610 A$=INKEY$:IF LEN(A$)=0 THEN 3610
3620 GOTO 3430
3630 REM
3640 CLOSE
3650 END
3660 SUB PRICING(RECNO) STATIC
3670 REM  UNIT PRICING DATA EDIT   SCREEN
3680 REM
3690 CLS:LOCATE 1,25:PRINT "U. S. ARMY CORPS OF ENGINEERS"
3700 LOCATE 2,30:PRINT "UNIT PRICING DATA"
3710 LOCATE 3,3:PRINT USING "RECORD NO. #####";RECNO
3720 LOCATE 5,7:PRINT "ITEM MARK INDEX"
3730 LOCATE 6,7:PRINT "PRIMARY CATEGORY"
3740 LOCATE 7,7:PRINT "MATERIAL SIZE"
3750 LOCATE 8,7:PRINT "PRICING UNIT OF MEASURE"
3760 LOCATE 9,7:PRINT "ALLOWABLE STRENGTH (KSI)"
3770 LOCATE 10,7:PRINT "ESTIMATE UNIT OF MEASURE"
3780 LOCATE 11,7:PRINT "ESTIMATE UNIT WEIGHT (WET OR DRY)"
3790 LOCATE 12,7:PRINT "FEDERAL STOCK NUMBER"
3800 LOCATE 13,7:PRINT "FABRICATED PART/USE DESCRIPTION"
3810 LOCATE 14,7:PRINT "MODULUS OF ELASTICITY (10**6)"
3820 LOCATE 15,7:PRINT "X-SECT AREA/DIAMETER/THICKNESS"
3830 LOCATE 16,7:PRINT "MOMENT OF INERTIA"
3840 LOCATE 17,7:PRINT "COEFF OF THERMAL EXPANSION (10**-6)"
3850 LOCATE 18,7:PRINT "ANCHOR SIZE (HELIX, INCHES)"
3860 LOCATE 19,7:PRINT "ITEM UNIT PRICE"
3870 LOCATE 20,7:PRINT "ERECTION TIME (MAN-HRS)"
3880 LOCATE 21,7:PRINT "DRAG COEFFICIENT (NET ONLY)"
3890 LOCATE 22,7:PRINT "NOT IN USE CURRENTLY"
3900 END SUB
```

5 CAMINP Routine

The input and edit program functions of the Camouflage Program are accomplished by the CAMINP routine. Two major program modules, CAMINP.BAS and CAMOSCRN.BAS, are linked together to provide these two program functions. The main calling routine is CAMINP which is CALLed from the mainline routine MAIN. The CAMINP routine then successively CALLs the input/edit routines for the general site information (SITE routine), the camouflage structure geometry (GEOM routine), the support member data (POLES routine), the netting data (NETS routine), the tension member data (CABLE routine), and the anchor data (ANCHOR routine). Each of these routines will be discussed in detail, and, following the discussions, source codes will be provided for each routine.

CAMINP Routine. The CAMINP routine is the main CALLing routine for the input and edit program functions of the Camouflage Program. The arguments to the CAMINP routine are the project name, PN$, the project revision number, RN$, the next available record in the project data file, NXRC, and the designated disk drive, DD$. The three data files required for program operation, the project data file (device #1), the screen field locations data file (DATA.LOC as device #2), and the screen field lengths (DATA.FLD as device #3) are OPENed and designated by device number before the CAMINP routine is CALLed. The CAMINP routine has three major functions to perform, as follows:

(1) Successively CALL the input/edit routines for the general site information, the camouflage structure geometry, the support member data, the netting data, the tension member data, and the anchor data,

(2) Close the field location and field length data files, and

(3) Copy the resultant project data file from the designated disk drive (DD$) back to the floppy disk drive, and erase the project data file from the designated disk drive.

Input/Edit Routine Calls. Each of the five input/edit screen routines will be called to allow the project data on each screen to be modified as desired by the user. The arguments of each routine are the project name (PN$), revision number (RN$), the next available record in the project data file (NXRC), and the termination keystroke indicator (KEYZ). At the conclusion of each of these screen routines, the project data file will be updated to record the data modifications that were accomplished. The sequential movement from one screen to the next can be altered by the termination keystroke provided on each screen. If the <ESC> key is used as a termination keystroke, a forward progression through the screens is provided. If the <F1> key is used, a backward progression through the screens is provided, thus allowing a review of the previous screens to ensure that the data provided are consistent in subsequent screens. The variable KEYZ is used to indicate the termination keystroke in each CALLed routine. If KEYZ is zero, the <ESC> key was used to terminate the input, and a forward progression is provided. If KEYZ is one, the <F1> key was used to terminate the input, and a backward progression is provided. The source codes for the screen input/edit routine CALLs and use of the KEYZ variable are provided below:

```
90 SUB CAMINP(PN$,RN$,NXRC,DD$)
100 REM
110 KEYZ=0:COLOR 15,1:CLS
120 REM
130 CALL SITE(PN$,RN$,NXRC,KEYZ)
140 CALL GEOM(PN$,RN$,NXRC,KEYZ):IF KEYZ=1 THEN 130
150 CALL POLES(PN$,RN$,NXRC,KEYZ):IF KEYZ=1 THEN 140
160 CALL NETS(PN$,RN$,NXRC,KEYZ):IF KEYZ=1 THEN 150
170 CALL CABLE(PN$,RN$,NXRC,KEYZ):IF KEYZ=1 THEN 160
180 CALL ANCHOR(PN$,RN$,NXRC,KEYZ):IF KEYZ=1 THEN 170
```

Each input/edit routine follows the same basic program logic. This logic is outlined in Figure 2 below. The steps of the logic process for each routine are as follows:

(1) Define the screen number and read the screen field location and field length data from the program data files,

(2) Check the project data file for existing project data related to the current input/edit screen,

(3) If no project data exist, load default data; if project data exist, load data from the project data file,

(4) Display the data and allow for the modification of all data items,

(5) Check the data for consistency where possible, displaying appropriate messages when inconsistencies are identified, and ensure data are corrected before program operations are continued, and

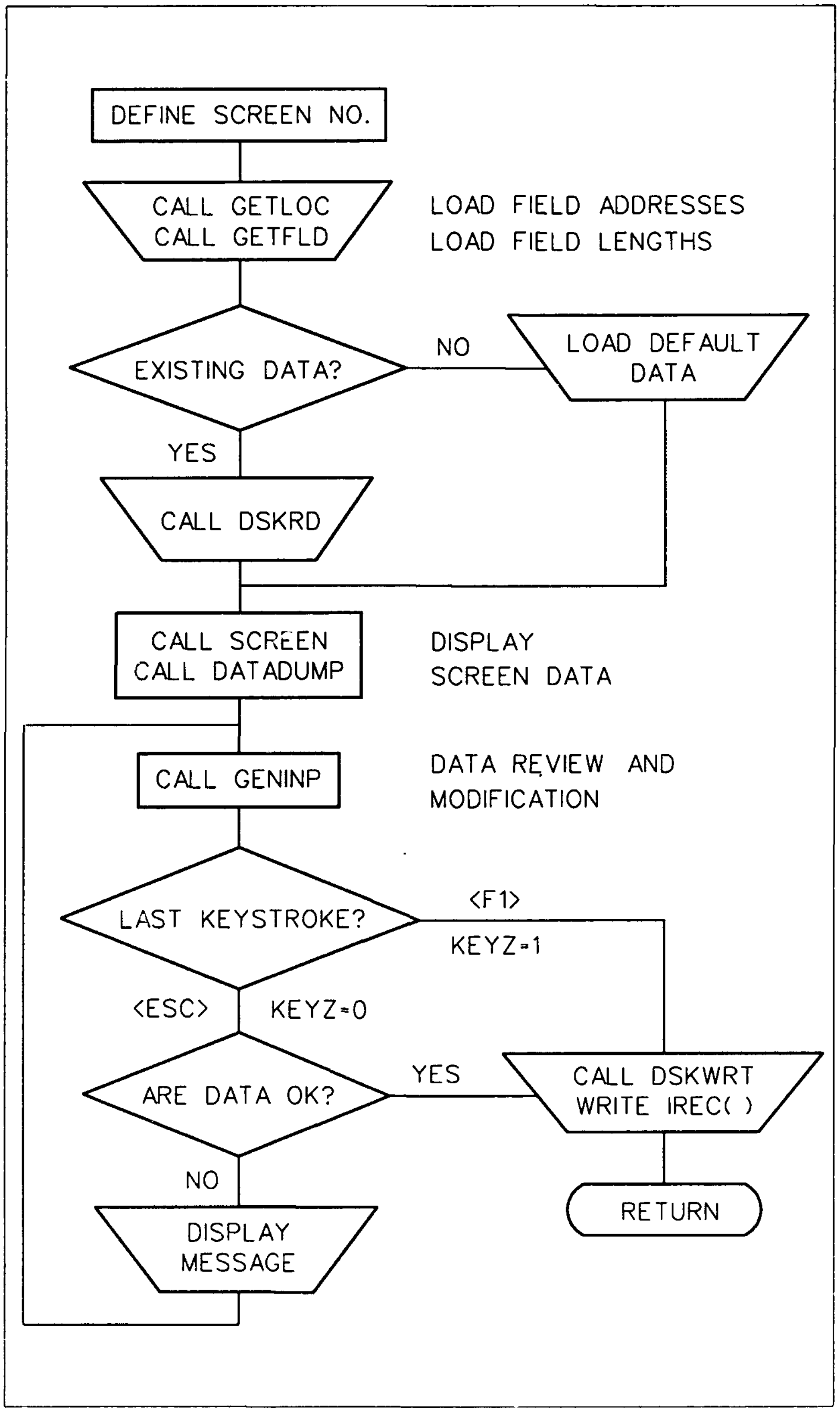

Figure 2. Camouflage input/edit routine logic

(6) Write the data to the project data file and update the addressing data both on the project data file and in memory.

General Site Description Routine. The first (NPR=1) input/edit routine called is the SITE routine which provides for the description of the general site conditions. These general site conditions include the soil characteristics, climatic data, wind conditions, and ice and snow load conditions. There are 11 data items (NAD(1)) associated with this screen. A description, data field locations, data field lengths, and default data of these data items are provided in Table 3 below. The auxiliary screen routine, SITESCRN, is used to provide the screen display for this routine.

Table 3
Input/Edit Data items for General Site Description Routine

Data Item	Field Location	Field Length	Default Data	Description
1	636	10	400.0	Soil cohesion (psf)
2	739	10	120.0	Soil unit weight (pcf)
3	858	10	25.0	Angle int friction (deg)
4	965	10	12.0	Boring diameter (in)
5	1354	10	0.10	Avg daily rainfall (in)
6	1457	10	50.0	Avg daily temp (deg F)
7	1539	10	10.0	Wind speed (mph)
8	1650	10		Avg sustained wind
9	1744	10		Gust wind speed
10	1854	10	2.0	Ice/snow load (psf)
11	2154	25		Terrain description

At the conclusion of the input/edit process (<ESC> termination keystroke), checks are made to ensure that values for the soil cohesion, soil unit weight, and soil angle of internal friction were provided. The soil cohesion must be specified in the range of 0-4000 psf,[1] the soil unit weight must be in the range of 20-150 pcf, and the angle of internal friction must be in the range of 0-50 degrees. If any of the three values are specified out of these ranges, an appropriate message is displayed, and

[1] A table of factors for converting non-SI units of measurement to SI units is presented on page vii.

control is returned to the screen display. The data checks are made each time the <ESC> termination keystroke is provided. The source codes for the SITE routine are provided below:

```
340 SUB SITE(PN$,RN$,NXRC,KEYZ)
480 REM
490 REM NLOC=11 FOR SITE DATA SCREEN
500 NPR=1:NLOC=NAD(NPR):CALL GETLOC(NPR,DUM())
510 FOR I=1 TO NAD(NPR):LPTS(I)=DUM(I):NEXT I
520 CALL GETFLD(NPR,DUM())
530 FOR I=1 TO NAD(NPR):FLD(I)=DUM(I):NEXT I
540 REM
550 FREC=IREC(NPR):IF FREC>0 THEN 640
560 REM
570 REM INITIALIZE SCREEN VALUES WITH DEFAULT DATA
580 FOR I=1 TO NLOC:VAR$(I)=" ":NEXT I
590 VAR$(1)=" 400.":VAR$(2)=" 120.":VAR$(3)=" 25.0":VAR$(4)
    =" 12.0"
600 VAR$(5)=" 0.10 ":VAR$(6)=" 50.0 ":VAR$(7)=" 10.0 "
610 VAR$(10)=" 2.0 "
620 GOTO 660
630 REM
640 CALL DSKRD(FREC,NDUM,DUM(),DUM$())
645 FOR I=1 TO NDUM:VAR$(I)=DUM$(I):NEXT I
650 REM
660 CALL SITESCRN(PN$,RN$)
665 IST=1:IEND=NLOC:CALL SCRNDUMP(IST,IEND)
670 LOCATE 24,2:PRINT "press <Esc> when input is complete . . .";
680 REM
690 REM ***  INPUT NEW DATA or OVER-WRITE old data  ***
700 CALL GENINP(IST,IEND,LAST)
710 IF LAST=59 THEN KEYZ=1:GOTO 940
720 IF LAST<>27 THEN 660
730 REM
740 REM DATA CHECKS FOR SITE SPECIFIC DATA
750 REM SOIL COHESION
760 A$=VAR$(1):CALL SCAN(A$,V(),NN,B$(),NW)
770 IF V(1)>0 AND V(1)<4000 THEN 840
780 B$(1)=" Invalid SOIL COHESION  #####.##  specified
    - TRY AGAIN"
790 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 790
800 COLOR 31,1,1:LOCATE 24,2:PRINT USING B$(1);V(1);
810 A$=INKEY$:IF LEN(A$)=0 THEN 810
820 GOTO 660
830 REM SOIL UNIT WEIGHT
840 A$=VAR$(2):CALL SCAN(A$,V(),NN,B$(),NW)
850 IF V(1)>20 AND V(1)<150 THEN 890
860 B$(1)=" Invalid SOIL UNIT WEIGHT  #####.##  specified
    - TRY AGAIN"
870 GOTO 790
```

```
880 REM INTERNAL ANGLE OF FRICTION
890 A$=VAR$(3):CALL SCAN(A$,V(),NN,B$(),NW)
900 IF V(1)>0 AND V(1)<50 THEN 940
910 B$(1)=" Invalid SOIL ANGLE OF INTERNAL FRICTION  ###.##
    specified "
920 GOTO 790
930 REM
940 FREC=IREC(NPR):IF FREC=0 THEN FREC=NXRC
950 CALL DSKWRT(FREC)
955 IF IREC(NPR)=0 THEN IREC(NPR)=NXRC:NXRC=FREC+1
960 A$=MKS$(NXRC):FOR I=1 TO 31:A$=A$+MKS$(IREC(I)):NEXT I
970 LSET AA$=A$:PUT #1,1
980 REM
990 END SUB
```

Structure Geometry Routine. The second (NPR=2) input/edit routine called is the GEOM which provides for the geometric description of the camouflage structure to be analyzed. These structure geometry data include the total length of the structure, bay spacings along the length of the structure, total width of the structure, minimum support spacing along the width dimension, support member height, exterior support guy condition, allowable sag in the tension members, and allowable displacement error of the support members. There are 27 data items (NAD(2)) associated with this screen. A description, data field locations, data field lengths, and default data of these data items are provided in Table 4 below. The auxiliary screen routine, GEOMSCRN, is used to provide the screen display for this routine.

Table 4
Input/Edit Data Items for Structure Geometry Routine

Data Item	Field Location	Field Length	Default Data	Description
1	648	10		Structure length (ft)
2-20 (2)	927-1167	2		Number of bays
3-21 (2)	932-1172	4		Bay spacing (ft)
22	1447	10		Structure width (ft)
23	1549	10	10.0	Min support spacing (ft)
24	1962	10	20.0	Structure height (ft)
25	2048	2	Y	Exterior support guyed
26	2150	10	2.0	Allowable sag (ft)
27	2260	10	0.10	Displacement error (in)

At the conclusion of the input/edit process (<ESC> termination keystroke), checks are made to ensure that the data are consistent. The structure length, structure width, minimum support spacing along the width, support height, maximum sag in the tension members, and maximum displacement error for the support members must be specified. The maximum displacement error must be at least 0.10 inches. The bay spacings are summed and the total of the bay spacings compared to the structure length. If any of these checks yield a discrepancy in the data, an appropriate message is displayed, and control is returned to the screen display. The data checks are made each time the <ESC> termination keystroke is provided. The source codes for the GEOM routine are provided below:

```
1000 SUB GEOM(PN$,RN$,NXRC,KEYZ)
1120 REM
1130 REM NLOC=27 FOR GEOMETRY SCREEN
1140 KEYZ=0:NPR=2:NLOC=NAD(NPR):CALL GETLOC(NPR,DUM())
1150 FOR I=1 TO NAD(NPR):LPTS(I)=DUM(I):NEXT I
1160 CALL GETFLD(NPR,DUM())
1170 FOR I=1 TO NAD(NPR):FLD(I)=DUM(I):NEXT I
1180 REM
1190 FREC=IREC(NPR):IF FREC>0 THEN 1260
1200 REM
1210 REM INITIALIZE SCREEN DATA ITEMS WITH DEFAULT DATA
1220 FOR I=1 TO NLOC:VAR$(I)=" ":NEXT I
1230 VAR$(23)=" 10.0 ":VAR$(24)=" 20.0 ":VAR$(25)="Y
     ":VAR$(26)=" 2.0 "
1240 VAR$(27)=" 0.10":GOTO 1280
1250 REM
1260 CALL DSKRD(FREC,NDUM,DUM(),DUM$())
1265 FOR I=1 TO NDUM:VAR$(I)=DUM$(I):NEXT I
1270 REM
1280 CALL GEOMSCRN(PN$,RN$)
1285 IST=1:IEND=NLOC:CALL SCRNDUMP(IST,IEND)
1290 LOCATE 24,2
1295 PRINT "press <Esc> for next screen   <F1> for previous
     screen";
1300 REM
1310 CALL GENINP(IST,IEND,LAST)
1320 IF LAST=59 THEN KEYZ=1:GOTO 1890
1330 IF LAST<>27 THEN 1280
1340 REM
1350 REM DATA CHECKS FOR CONSISTENCY OF GEOMETRY
1360 A$=VAR$(1):CALL SCAN(A$,V(),NN,B$(),NW):SLENG=V(1)
1370 IF SLENG>0 THEN 1450
1380 B$(1)=" Invalid Structure Length  ####.###  was specified
     - TRY AGAIN"
1390 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 1390
1400 COLOR 31,1,1:LOCATE 24,2:PRINT USING B$(1);SLENG;
1410 A$=INKEY$:IF LEN(A$)=0 THEN 1410
```

```
1420 COLOR 15,1,1:GOTO 1280
1430 REM
1440 REM CHECK BAY SPACINGS
1450 SPAN=0!:NBAY=0:FOR IZ=2 TO 20 STEP 2
1460 A$=VAR$(IZ):CALL SCAN(A$,V(),NN,B$(),NW):IF V(1)<=0 THEN 1490
1470 NBY=V(1):A$=VAR$(IZ+1):CALL SCAN(A$,V(),NN,B$(),NW)
1480 SPAN=SPAN+NBY*V(1):NBAY=NBAY+NBY
1490 NEXT IZ
1500 IF ABS(SLENG-SPAN)<.01 THEN 1570
1510 B$(1)=" Sum of Bay Spacings  ####.###  <>  Structure
     Length  ####.###"
1520 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 1520
1530 COLOR 31,1,1:LOCATE 24,2:PRINT USING B$(1);SPAN,SLENG;
1540 GOTO 1410
1550 REM
1560 REM CHECK STRUCTURE WIDTH
1570 A$=VAR$(22):CALL SCAN(A$,V(),NN,B$(),NW):IF V(1)>0 THEN 1620
1580 B$(1)=" Invalid Structure Width  ####.###  was specified
     - TRY AGAIN"
1590 SLENG=V(1):GOTO 1390
1600 REM
1610 REM CHECK MIN POLE SPACING
1620 A$=VAR$(23):CALL SCAN(A$,V(),NN,B$(),NW):IF V(1)>0 THEN 1640
1625 B$(1)=" Invalid Min Pole Spacing  ####.###  was specified
     - TRY AGAIN"
1630 MINSPC=V(1):SLENG=MINSPC:GOTO 1390
1635 REM CHECK NUMBER SUPPORT MEMBERS REQUIRED (MAX OF 64)
1640 NWIDE=INT(WIDE/MINSPC)+1:NPOLE=NWIDE*(NBAY+1)
1645 IF NPOLE<=64 THEN 1670
1650 B$(1)=" Min Pole Spacing will exceed Max Supports Allowed
     (64)  ###"
1655 SLENG=NPOLE:GOTO 1390
1660 REM
1665 REM CHECK EXT POLE HEIGHT
1670 A$=VAR$(24):CALL SCAN(A$,V(),NN,B$(),NW):IF V(1)>0 THEN 1720
1680 B$(1)=" Invalid Ext Pole Height  ####.###  was specified
     - TRY AGAIN"
1690 SLENG=V(1):GOTO 1390
1700 REM
1710 REM CHECK EXT POLE GUY SPECIFICATION (Y/N)
1720 A$=VAR$(25):IF MID$(A$,1,1)="Y" OR MID$(A$,1,1)="y" THEN 1800
1730 IF MID$(A$,1,1)="N" OR MID$(A$,1,1)="n" THEN 1800
1740 B$(1)=" Invalid Ext Pole Guy  \\  was specified
     - TRY AGAIN"
1750 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 1750
1760 COLOR 31,1,1:LOCATE 24,2:PRINT USING B$(1);A$;
1770 GOTO 1410
1780 REM
1790 REM CHECK MAX SAG ALLOWED IN CABLES
1800 A$=VAR$(26):CALL SCAN(A$,V(),NN,B$(),NW):IF V(1)>0 THEN 1850
```

```
1810 B$(1)=" Invalid Maximum Cable Sag  ####.###  was specified
     - TRY AGAIN"
1820 SLENG=V(1):GOTO 1390
1830 REM
1840 REM CHECK MAX ALLOW DISPLACE ERROR (MUST BE > THAN 0.10 IN)
1850 A$=VAR$(27):CALL SCAN(A$,V(),NN,B$(),NW):IF V(1)>=.1
     THEN 1890
1860 B$(1)=" Allowable Displacement Error  ##.####  must be
     > 0.10 inches "
1870 SLENG=V(1):GOTO 1390
1880 REM
1890 FREC=IREC(NPR):IF FREC=0 THEN FREC=NXRC
1900 CALL DSKWRT(FREC)
1905 IF IREC(NPR)=0 THEN IREC(NPR)=NXRC:NXRC=FREC+1
1910 A$=MKS$(NXRC):FOR I=1 TO 31:A$=A$+MKS$(IREC(I)):NEXT I
1920 LSET AA$=A$:PUT #1,1
1940 REM
1950 END SUB
```

Support Member Characteristics Routine. The third and fourth (NPR=3 and 4) input/edit screens displayed are provided from the POLES routine, which provides for the material characteristics of the support member for the structure. Up to 30 support members may be included on these two screens of input, but only one may be selected for each program execution. The support member characteristics include the use indicator, six-character name of the member, member length, allowable strength, modulus of elasticity, cross-sectional area, moment of inertia, unit price, and a 15-character description. There are 9 data items (NAD(3) and NAD(4)) associated with each support member, and each screen may contain a maximum of 15 support members. A total of 135 data items can be provided on each of the two screens provided for the support member data. A description, data field locations, data field lengths, and default data of the nine data items for a single support member are provided in Table 5 below. The data for the other 14 support members contained on the screen will be identical to those data in Table 5 except that the field location will be increased by 100 for each additional support member displayed (1002, 1004, 1011, 1020, 1029, 1038, 1047, 1056, and 1065 for the second; 1102, etc., for the third and so on). The auxiliary screen routine, POLSCRN, is used to provide the screen display for this routine.

The initial default data for each support member are provided from the program pricing data (Chapter 4 Pricing Data). During the input program function, the pricing data file is opened, and up to 30 support members are extracted and stored in the POLNAME$() array. These pricing data items are marked with a **P** use code. If the pricing data file is empty, or if no data items are found with a **P** use code, the initial display of the screen will be empty. In this case, the user will have to provide each of the nine data characteristics for the support member to be used. If the desired support member is not displayed, the user may add the support member at the end of the display and provide each of the nine data characteristics for

Table 5
Input/Edit Data Items for Support Member Characteristics Routine

Data Item	Field Location	Field Length	Default Data	Description
1	902	1	N	Use indicator (Y/N)
2	904	6		Name of member
3	911	8		Length of member (ft)
4	920	8		Allowable strength (ksi)
5	929	8		Modulus of elasticity
6	938	8		Cross-section area
7	947	8		Moment of inertia
8	956	8		Unit price ($/ea)
9	965	15		Description

that member. During subsequent edit program functions, the same material data will be extracted from the project data file, and no additional searching of the pricing data file will be attempted. At the conclusion of the input/edit process (<ESC> termination keystroke), each data item for the support member selected will be checked to ensure that a nonzero value has been provided. The final data check ensures that only one support member was selected. If any of these checks indicate an inconsistent data set, an appropriate message is displayed, and control is returned to the screen display. The data checks are made each time the <ESC> termination keystroke is provided. The source codes for the POLES routine are provided below:

```
1960 SUB POLES(PN$,RN$,NXRC,KEYZ)
1970 REM
2100 DIM POLNAME$(30)
2110 NPOL=0:NS=BEGAD(27):IF NS=0 THEN 2170
2120 NE=BEGAD(28)-1
2130 FOR I=NS TO NE:GET #5,I+79
2140 NPOL=NPOL+1:POLNAME$(NPOL)=PRD$(1)
2150 NEXT I
2160 REM
2170 KEYZ=0:NPR=3:NYES=0
2180 CALL GETLOC(NPR,DUM()):NLOC=NAD(NPR)*15
2190 FOR I=1 TO NAD(NPR):LPTS(I)=DUM(I):NEXT I
2200 CALL GETFLD(NPR,DUM())
2205 FOR I=1 TO NAD(NPR):FLD(I)=DUM(I):NEXT I
2210 FOR I=2 TO 15:IJ=(I-1)*NAD(NPR):FOR J=1 TO NAD(NPR)
2220 LPTS(IJ+J)=LPTS(J)+100*(I-1):DUM(IJ+J)=DUM(J):FLD(IJ+J)
     =DUM(J)
```

```
2230 NEXT J:NEXT I
2240 REM
2250 FREC=IREC(NPR):IF FREC>0 THEN 2400
2260 REM
2270 NSTRT=1:IF NPR=4 THEN NSTRT=16
2280 NEND=NSTRT+14
2290 FOR I=NSTRT TO NEND:IJ=(I-NSTRT)*NAD(NPR)
2300 FOR J=1 TO NAD(NPR):VAR$(IJ+J)=" ":NEXT J
2310 IF I>NPOL THEN 2380
2320 VAR$(IJ+1)="N"
2330 VAR$(IJ+2)=POLNAME$(I):CALL LOOKPRIC(POLNAME$(I),PRST$)
2340 VAR$(IJ+3)=STR$(CVS(PRD$(3))):VAR$(IJ+4)=STR$(CVS(PRD$(5)))
2350 VAR$(IJ+5)=STR$(CVS(PRD$(10))):VAR$(IJ+6)=STR$(CVS(PRD$(11)))
2360 VAR$(IJ+7)=STR$(CVS(PRD$(12))):VAR$(IJ+8)=STR$(CVS(PRD$(15)))
2370 VAR$(IJ+9)=MID$(PRD$(9),1,15)
2380 NEXT I:GOTO 2420
2390 REM
2400 CALL DSKRD(FREC,NDUM,DUM(),DUM$())
2405 FOR I=1 TO NDUM:VAR$(I)=DUM$(I):NEXT I
2410 REM
2420 CALL POLSCRN(PN$,RN$)
2425 IST=1:IEND=NLOC:CALL SCRNDUMP(IST,IEND)
2430 LOCATE 24,2
2435 PRINT "press <Esc> for next screen   <F1> for previous
     screen";
2440 REM
2450 CALL GENINP(IST,IEND,LAST)
2460 IF LAST=59 THEN KEYZ=1:GOTO 2690
2470 REM
2480 REM  CHECK FOR ONLY ONE SPECIFIED MATERIAL HERE
2490 FOR I=1 TO 15:IJ=(I-1)*NAD(NPR)
2500 IF VAR$(IJ+1)="Y" OR VAR$(IJ+1)="y" THEN
2510      REM
2520      REM INSURE THAT ALL DATA ITEMS ARE SPECIFIED
2530      FOR JZ=3 TO 8:A$=VAR$(IJ+JZ):CALL SCAN(A$,V(),NN,
          B$(),NW)
2540      IF V(1)>0 THEN 2590
2550      B$(1)=" SUPPORT MEMBER ##:  All data items must
          be specified"
2560      IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 2560
2570      COLOR 31,1,1:LOCATE 24,2:PRINT USING B$(1);I;
2580      GOTO 2660
2590      NEXT JZ:NYES=NYES+1
2600 ENDIF
2610 NEXT I:IF NPR=3 THEN 2690
2620 IF NYES=1 THEN 2690
2630 B$(1)=" Only one (1) POLE material should be selected "
2640 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 2640
2650 COLOR 31,1,1:LOCATE 24,2:PRINT B$(1);
2660 A$=INKEY$:IF LEN(A$)=0 THEN 2660
```

```
2670 COLOR 15,1,1:GOTO 2420
2680 REM
2690 FREC=IREC(NPR):IF FREC=0 THEN FREC=NXRC
2700 CALL DSKWRT(FREC)
2705 IF IREC(NPR)=0 THEN IREC(NPR)=NXRC:NXRC=FREC+1
2710 A$=MKS$(NXRC):FOR I=1 TO 31:A$=A$+MKS$(IREC(I)):NEXT I
2720 LSET AA$=A$:PUT #1,1
2730 IF LAST=59 THEN 2750
2740 IF NPR=3 THEN NPR=NPR+1:GOTO 2180
2750 END SUB
```

Netting Characteristics Routine. The fifth and sixth (NPR=5 and 6) input/edit screens displayed are provided from the NETS routine, which provides for the material characteristics of the netting for the structure. Up to 30 netting types may be included on these two screens of input, but only one may be selected for each program execution. The netting characteristics include the use indicator, six-character name of the net, unit size of each netting piece, thickness, allowable strength, unit weight, drag coefficient, unit price, and a 15-character description. Nine data items (NAD(5) and NAD(6)) are associated with each netting type, and each screen may contain a maximum of 15 netting types. A total of 135 data items can be provided on each of the two screens provided for the netting data. A description, data field locations, data field lengths, and default data of the nine data items for a single netting type are provided in Table 6 below. The data for the other 14 netting types contained on the screen will be identical to those data in Table 6 except that the field location will be increased by 100 for each additional netting type displayed. The auxiliary screen routine, NETSCRN, is used to provide the screen display for this routine.

The initial default data for each netting type are provided from the program pricing data (Chapter 4 Pricing Data). During the input program function, the pricing data file is opened, and up to 30 netting types are extracted and stored in the NETNAME$() array. These pricing data items are marked with an N use code. If the pricing data file is empty, or if no data items are found with an N use code, the initial display of the screen will be empty. In this case, the user will have to provide each of the nine data characteristics for the netting type to be used. If the desired netting type is not displayed, the user may add the netting type at the end of the display and provide each of the nine data characteristics for that type. During subsequent edit program functions, the same material data will be extracted from the project data file, and no additional searching of the pricing data file will be attempted. At the conclusion of the input/edit process (<ESC> termination keystroke), each data item for the netting type selected will be checked to ensure that a nonzero value has been provided. The final data check ensures that only one netting type was selected. If any of these checks indicate an inconsistent data set, an appropriate message is displayed, and control is returned to the screen display. The data checks are made each time the <ESC> termination

Table 6
Input/Edit Data Items for Netting Characteristics Routine

Data Item	Field Location	Field Length	Default Data	Description
1	902	1	N	Use indicator (Y/N)
2	904	6		Name of netting type
3	911	8		Unit size (sq ft)
4	920	8		Net thickness (in)
5	929	8		Allowable strength
6	938	8		Unit weight (lbs/sq ft)
7	947	8		Drag coefficient
8	956	8		Unit price ($/sq ft)
9	965	15		Description

keystroke is provided. The source codes for the NETS routine are provided below:

```
2760 SUB NETS(PN$,RN$,NXRC,KEYZ)
2770 REM
2900 DIM NETNAME$(30)
2910 NNET=0:NS=BEGAD(25):IF NS=0 THEN 2960
2920 NE=BEGAD(26)-1
2930 FOR I=NS TO NE:GET #5,I+79
2940 NNET=NNET+1:NETNAME$(NNET)=PRD$(1)
2950 NEXT I
2960 KEYZ=0:NPR=5:NYES=0
2970 CALL GETLOC(NPR,DUM()):NLOC=NAD(NPR)*15
2980 FOR I=1 TO NAD(NPR):LPTS(I)=DUM(I):NEXT I
2990 CALL GETFLD(NPR,DUM())
2995 FOR I=1 TO NAD(NPR):FLD(I)=DUM(I):NEXT I
3000 FOR I=2 TO 15:IJ=(I-1)*NAD(NPR):FOR J=1 TO NAD(NPR)
3010 LPTS(IJ+J)=LPTS(J)+100*(I-1):DUM(IJ+J)=DUM(J):FLD(IJ+J)
     =DUM(J)
3020 NEXT J:NEXT I
3030 REM
3040 FREC=IREC(NPR):IF FREC>0 THEN 3190
3050 REM
3060 NSTRT=1:IF NPR=6 THEN NSTRT=16
3070 NEND=NSTRT+14
3080 FOR I=NSTRT TO NEND:IJ=(I-NSTRT)*NAD(NPR)
3090 FOR J=1 TO NAD(NPR):VAR$(IJ+J)=" ":NEXT J
3100 IF I>NNET THEN 3170
```

```
3110 VAR$(IJ+1)="N"
3120 VAR$(IJ+2)=NETNAME$(I):CALL LOOKPRIC(NETNAME$(I),PRST$)
3130 VAR$(IJ+3)=STR$(CVS(PRD$(3))):VAR$(IJ+4)=STR$(CVS(PRD$(11)))
3140 VAR$(IJ+5)=STR$(CVS(PRD$(5))):VAR$(IJ+6)=STR$(CVS(PRD$(7)))
3150 VAR$(IJ+7)=STR$(CVS(PRD$(17))):VAR$(IJ+8)=STR$(CVS(PRD$(15)))
3160 VAR$(IJ+9)=MID$(PRD$(9),1,15)
3170 NEXT I:GOTO 3210
3180 REM
3190 CALL DSKRD(FREC,NDUM,DUM(),DUM$())
3195 FOR I=1 TO NDUM:VAR$(I)=DUM$(I):NEXT I
3200 REM
3210 CALL NETSCRN(PN$,RN$)
3215 IST=1:IEND=NLOC:CALL SCRNDUMP(IST,IEND)
3220 LOCATE 24,2
3225 PRINT "press <Esc> for next screen   <F1> for previous
     screen";
3230 REM
3240 CALL GENINP(IST,IEND,LAST)
3250 IF LAST=59 THEN KEYZ=1:GOTO 3480
3260 REM
3270 REM CHECK FOR ONLY ONE MATERIAL SPECIFIED
3280 FOR I=1 TO 15:IJ=(I-1)*NAD(NPR)
3290 IF VAR$(IJ+1)="Y" OR VAR$(IJ+1)="y" THEN
3300      REM
3310      REM INSURE THAT ALL DATA ITEMS ARE SPECIFIED
3320      FOR JZ=3 TO 8:A$=VAR$(IJ+JZ):CALL SCAN(A$,V(),
          NN,B$(),NW)
3330      IF V(1)>0 THEN 3380
3340      B$(1)=" NETTING MATERIAL ##:  All data items must
          be specified"
3350      IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 3350
3360      COLOR 31,1,1:LOCATE 24,2:PRINT USING B$(1);I;
3370      GOTO 3450
3380      NEXT JZ:NYES=NYES+1
3390 ENDIF
3400 NEXT I:IF NPR=5 THEN 3480
3410 IF NYES=1 THEN 3480
3420 B$(1)=" Only one (1) NETTING material should be selected "
3430 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 3430
3440 COLOR 31,1,1:LOCATE 24,2:PRINT B$(1);
3450 A$=INKEY$:IF LEN(A$)=0 THEN 3450
3460 COLOR 15,1,1:GOTO 3210
3470 REM
3480 FREC=IREC(NPR):IF FREC=0 THEN FREC=NXRC
3490 CALL DSKWRT(FREC)
3495 IF IREC(NPR)=0 THEN IREC(NPR)=NXRC:NXRC=FREC+1
3500 A$=MKS$(NXRC):FOR I=1 TO 31:A$=A$+MKS$(IREC(I)):NEXT I
3510 LSET AA$=A$:PUT #1,1
3520 IF LAST=59 THEN 3540
3530 IF NPR=5 THEN NPR=NPR+1:GOTO 2970
```

```
3540 END SUB
```

Tension Member Characteristics Routine. The seventh and eighth (NPR=7 and 8) input/edit screens displayed are provided from the CABLES routine, which provides for the material characteristics of the tension member for the structure. Up to 30 tension members may be included on these two screens of input, but only one may be selected for each program execution. The tension member characteristics include the use indicator, six-character name of the member, allowable strength, modulus of elasticity, coefficient of thermal expansion, cross-sectional area, unit weight, unit price, and a 15-character description. Nine data items (NAD(7) and NAD(8)) are associated with each tension member, and each screen may contain a maximum of 15 tension members. A total of 135 data items can be provided on each of the two screens provided for the tension member data. A description, data field locations, data field lengths, and default data of the nine data items for a single tension member are provided in Table 7 below. The data for the other 14 tension members contained on the screen will be identical to those data in Table 7 except that the field location will be increased by 100 for each additional tension member displayed. The auxiliary screen routine, TENSCRN, is used to provide the screen display for this routine.

The initial default data for each tension member are provided from the program pricing data (Chapter 4 Pricing Data). During the input program function, the pricing data file is opened, and up to 30 tension members are extracted and stored in the TENNAME$() array. These pricing data items

Table 7
Input/Edit Data Items for Tension Member Characteristics Routine

Data Item	Field Location	Field Length	Default Data	Description
1	902	1	N	Use indicator (Y/N)
2	904	6		Name of member
3	911	8		Allowable strength
4	920	8		Modulus of elasticity
5	929	8		Coef of thermal expans
6	938	8		Cross-section area
7	947	8		Unit weight (lbs/ft)
8	956	8		Unit price ($/ft)
9	965	15		Description

are marked with a **C** use code. If the pricing data file is empty, or if no data items are found with a **C** use code, the initial display of the screen will be empty. In this case, the user will have to provide each of the nine data characteristics for the tension member to be used. If the desired tension member is not displayed, the user may add the tension member at the end of the display and provide each of the nine data characteristics for that member. During subsequent edit program functions, the same material data will be extracted from the project data file, and no additional searching of the pricing data file will be attempted. At the conclusion of the input/edit process (<ESC> termination keystroke), each data item for the tension member selected will be checked to ensure that a nonzero value has been provided. The final data check ensures that only one tension member was selected. If any of these checks indicate an inconsistent data set, an appropriate message is displayed, and control is returned to the screen display. The data checks are made each time the <ESC> termination keystroke is provided. The source codes for the CABLE routine are provided below:

```
3550 SUB CABLE(PN$,RN$,NXRC,KEYZ)
3560 REM
3690 DIM TENNAME$(30)
3700 NTEN=0:NS=BEGAD(14):IF NS=0 THEN 3750
3710 NE=BEGAD(15)-1
3720 FOR I=NS TO NE:GET #5,I+79
3730 NTEN=NTEN+1:TENNAME$(NTEN)=PRD$(1)
3740 NEXT I
3750 KEYZ=0:NPR=7:NYES=0
3760 CALL GETLOC(NPR,DUM()):NLOC=NAD(NPR)*15
3770 FOR I=1 TO NAD(NPR):LPTS(I)=DUM(I):NEXT I
3780 CALL GETFLD(NPR,DUM())
3785 FOR I=1 TO NAD(NPR):FLD(I)=DUM(I):NEXT I
3790 FOR I=2 TO 15:IJ=(I-1)*NAD(NPR):FOR J=1 TO NAD(NPR)
3800 LPTS(IJ+J)=LPTS(J)+100*(I-1):DUM(IJ+J)=DUM(J):FLD(IJ+J)
     =DUM(J)
3810 NEXT J:NEXT I
3820 REM
3830 FREC=IREC(NPR):IF FREC>0 THEN 3980
3840 REM
3850 NSTRT=1:IF NPR=8 THEN NSTRT=16
3860 NEND=NSTRT+14
3870 FOR I=NSTRT TO NEND:IJ=(I-NSTRT)*NAD(NPR)
3880 FOR J=1 TO NAD(NPR):VAR$(IJ+J)=" ":NEXT J
3890 IF I>NTEN THEN 3960
3900 VAR$(IJ+1)="N"
3910 VAR$(IJ+2)=TENNAME$(I):CALL LOOKPRIC(TENNAME$(I),PRST$)
3920 VAR$(IJ+3)=STR$(CVS(PRD$(5))):VAR$(IJ+4)=STR$(CVS(PRD$(10)))
3930 VAR$(IJ+5)=STR$(CVS(PRD$(13))):VAR$(IJ+6)=STR$(CVS(PRD$(11)))
3940 VAR$(IJ+7)=STR$(CVS(PRD$(7))):VAR$(IJ+8)=STR$(CVS(PRD$(15)))
3950 VAR$(IJ+9)=MID$(PRD$(9),1,15)
3960 NEXT I:GOTO 4000
```

```
3970 REM
3980 CALL DSKRD(FREC,NDUM,DUM(),DUM$())
3985 FOR I=1 TO NDUM:VAR$(I)=DUM$(I):NEXT I
3990 REM
4000 CALL TENSCRN(PN$,RN$)
4005 IST=1:IEND=NLOC:CALL SCRNDUMP(IST,IEND)
4010 LOCATE 24,2
4015 PRINT "press <Esc> for next screen   <F1> for previous
     screen";
4020 REM
4030 CALL GENINP(IST,IEND,LAST)
4040 IF LAST=59 THEN KEYZ=1:GOTO 4270
4050 REM
4060 REM CHECK FOR ONLY ONE MATERIAL SPECIFIED
4070 FOR I=1 TO 15:IJ=(I-1)*NAD(NPR)
4080 IF VAR$(IJ+1)="Y" OR VAR$(IJ+1)="y" THEN
4090      REM
4100      REM INSURE THAT ALL DATA ITEMS ARE SPECIFIED
4110      FOR JZ=3 TO 8:A$=VAR$(IJ+JZ):CALL SCAN(A$,V(),NN,
          B$(),NW)
4120      IF V(1)>0 THEN 4170
4130      B$(1)=" TENSION MEMBER ##:  All data items must
          be specified"
4140      IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 4140
4150      COLOR 31,1,1:LOCATE 24,2:PRINT USING B$(1);I;
4160      GOTO 4240
4170      NEXT JZ:NYES=NYES+1
4180 ENDIF
4190 NEXT I:IF NPR=7 THEN 4270
4200 IF NYES=1 THEN 4270
4210 B$(1)=" Only one (1) TENSION MEMBER material should
     be selected "
4220 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 4220
4230 COLOR 31,1,1:LOCATE 24,2:PRINT B$(1);
4240 A$=INKEY$:IF LEN(A$)=0 THEN 4240
4250 COLOR 15,1,1:GOTO 4000
4260 REM
4270 FREC=IREC(NPR):IF FREC=0 THEN FREC=NXRC
4280 CALL DSKWRT(FREC)
4285 IF IREC(NPR)=0 THEN IREC(NPR)=NXRC:NXRC=FREC+1
4290 A$=MKS$(NXRC):FOR I=1 TO 31:A$=A$+MKS$(IREC(I)):NEXT I
4300 LSET AA$=A$:PUT #1,1
4310 IF LAST=59 THEN 4330
4320 IF NPR=7 THEN NPR=NPR+1:GOTO 3760
4330 END SUB
```

Anchor Characteristics Routine. The ninth and tenth (NPR=9 and 10) input/edit screens displayed are provided from the ANCHOR routine, which provides for the material characteristics of the anchors for the guyed structure. Up to 30 anchors may be included on these two screens of input, but only five may be selected for each program execution. The critical data for these five anchors are stored in the labeled COMMON block ANCHORS. The anchor characteristics include the use indicator, six-character name of the anchor, rod length, allowable strength, modulus of elasticity, rod diameter, helix diameter, unit price, and a 15-character description. Nine data items (NAD(9) and NAD(10)) are associated with each anchor, and each screen may contain a maximum of 15 anchors. A total of 135 data items can be provided on each of the two screens provided for the anchor data. A description, data field locations, data field lengths, and default data of the nine data items for a single anchor are provided in Table 8 below. The data for the other 14 anchors contained on the screen will be identical to those data in Table 8 except that the field location will be increased by 100 for each additional anchor displayed. The auxiliary screen routine, ANCSCRN, is used to provide the screen display for this routine.

The initial default data for each anchor are provided from the program pricing data (Chapter 4 Pricing Data). During the input program function, the pricing data file is opened, and up to 30 anchors are extracted and stored in the ANCNAME$() array. These pricing data items are marked with an A use code. If the pricing data file is empty, or if no data items are found with an A use code, the initial display of the screen will be empty. In this case, the user will have to provide each of the nine data characteristics for the anchor to be used. If the desired anchor is not displayed, the user may add the anchor at the end of the display and provide each of the nine data characteristics for that anchor. During subsequent edit program functions, the same material data will be extracted from the project data file, and no additional searching of the pricing data file will

Table 8
Input/Edit Data Items for anchor Characteristics Routine

Data Item	Field Location	Field length	Default Data	Description
1	902	1	N	Use indicator (Y/N)
2	904	6		Name of anchor
3	911	8		Rod length (in)
4	920	8		Allowable strength (ksi)
5	929	8		Modulus of elasticity
6	938	8		Rod diameter (in)
7	947	8		Helix diameter (in)
8	956	8		Unit price ($/ea)
9	965	15		Description

be attempted. At the conclusion of the input/edit process (<ESC> termination keystroke), each data item for the anchors selected will be checked to ensure that a nonzero value has been provided. The final data check ensures that no more than five anchors were selected. If any of these checks indicate an inconsistent data set, an appropriate message is displayed, and control is returned to the screen display. The data checks are made each time the <ESC> termination keystroke is provided. The source codes for the ANCHOR routine are provided below:

```
4340 SUB ANCHOR(PN$,RN$,NXRC,KEYZ)
4350 REM

4480 DIM ANCNAME$(30)
4490 NANC=0:NS=BEGAD(12):IF NS=0 THEN 4540
4500 NE=BEGAD(13)-1
4510 FOR I=NS TO NE:GET #5,I+79
4520 NANC=NANC+1:ANCNAME$(NANC)=PRD$(1)
4530 NEXT I
4540 KEYZ=0:NPR=9:NYES=0
4550 CALL GETLOC(NPR,DUM()):NLOC=NAD(NPR)*15
4560 FOR I=1 TO NAD(NPR):LPTS(I)=DUM(I):NEXT I
4570 CALL GETFLD(NPR,DUM())
4575 FOR I=1 TO NAD(NPR):FLD(I)=DUM(I):NEXT I
4580 FOR I=2 TO 15:IJ=(I-1)*NAD(NPR):FOR J=1 TO NAD(NPR)
4590 LPTS(IJ+J)=LPTS(J)+100*(I-1):DUM(IJ+J)=DUM(J):FLD(IJ+J)
     =DUM(J)
4600 NEXT J:NEXT I
4610 REM
4620 FREC=IREC(NPR):IF FREC>0 THEN 4770
4630 REM
4640 NSTRT=1:IF NPR=10 THEN NSTRT=16
4650 NEND=NSTRT+14
4660 FOR I=NSTRT TO NEND:IJ=(I-NSTRT)*NAD(NPR)
4670 FOR J=1 TO NAD(NPR):VAR$(IJ+J)=" ":NEXT J
4680 IF I>NANC THEN 4750
4690 VAR$(IJ+1)="N"
4700 VAR$(IJ+2)=ANCNAME$(I):CALL LOOKPRIC(ANCNAME$(I),PRST$)
4710 VAR$(IJ+3)=STR$(CVS(PRD$(3))):VAR$(IJ+4)=STR$(CVS(PRD$(5)))
4720 VAR$(IJ+5)=STR$(CVS(PRD$(10))):VAR$(IJ+6)=STR$(CVS(PRD$(11)))
4730 VAR$(IJ+7)=STR$(CVS(PRD$(14))):VAR$(IJ+8)=STR$(CVS(PRD$(15)))
4740 VAR$(IJ+9)=MID$(PRD$(9),1,15)
4750 NEXT I:GOTO 4790
4760 REM
4770 CALL DSKRD(FREC,NDUM,DUM(),DUM$())
4775 FOR I=1 TO NDUM:VAR$(I)=DUM$(I):NEXT I
4780 REM
4790 CALL ANCSCRN(PN$,RN$)
4795 IST=1:IEND=NLOC:CALL SCRNDUMP(IST,IEND)
4800 LOCATE 24,2
```

```
4805 PRINT "press <Esc> for next screen   <F1> for previous
     screen";
4810 REM
4820 CALL GENINP(IST,IEND,LAST)
4830 IF LAST=59 THEN KEYZ=1:GOTO 5100
4840 REM
4850 REM CHECK FOR UP TO FIVE ANCHORS SPECIFIED
4860 FOR I=1 TO 15:IJ=(I-1)*NAD(NPR)
4870 IF VAR$(IJ+1)="Y" OR VAR$(IJ+1)="y" THEN
4880      REM
4890      REM INSURE THAT ALL DATA ITEMS ARE SPECIFIED
4900      FOR JZ=3 TO 8:A$=VAR$(IJ+JZ):CALL SCAN(A$,V(),NN,
          B$(),NW)
4910      IF V(1)>0 THEN 4960
4920      B$(1)=" ANCHOR ##:  All data items must be specified"
4930      IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 4930
4940      COLOR 31,1,1:LOCATE 24,2:PRINT USING B$(1);I;
4950      GOTO 5030
4960      NEXT JZ:NYES=NYES+1
4970 ENDIF
4980 NEXT I:IF NPR=9 THEN 5110
4990 IF NYES>0 THEN 5060
5000 B$(1)=" At least one (1) ANCHOR material must be selected "
5010 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 5010
5020 COLOR 31,1,1:LOCATE 24,2:PRINT B$(1);
5030 A$=INKEY$:IF LEN(A$)=0 THEN 5030
5040 COLOR 15,1,1:GOTO 4790
5050 REM
5060 IF NYES<6 THEN 5110
5070 B$(1)=" No more than five (5) ANCHOR materials may
     be selected "
5080 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 5080
5090 GOTO 5020
5100 REM
5110 FREC=IREC(NPR):IF FREC=0 THEN FREC=NXRC
5120 CALL DSKWRT(FREC)
5125 IF IREC(NPR)=0 THEN IREC(NPR)=NXRC:NXRC=FREC+1
5130 A$=MKS$(NXRC):FOR I=1 TO 31:A$=A$+MKS$(IREC(I)):NEXT I
5140 LSET AA$=A$:PUT #1,1
5150 IF LAST=59 THEN 5170
5160 IF NPR=9 THEN NPR=NPR+1:GOTO 4550
5170 END SUB
```

Close Program Data Files. At the conclusion of the input/edit program functions when program control has been returned from the ANCHOR routine, the CAMINP routine then closes the two data files used to extract the field location data and field length data. The two data files were OPENed as device numbers 2 and 3, respectively, so the CLOSE function is called for each device. The source code is provided below:

```
190 CLOSE #2:CLOSE #3
```

Copy Project Data to Floppy Disk. The final phase of CAMINP routine operations involves the transfer of the project data file from the designated disk drive (DD$) to the floppy disk drive and the erasure of the project data file from the designated disk drive. These functions are performed to ensure that the project data are retained on an external storage medium and to avoid the designated disk drive (hard disk possibly) from becoming cluttered with project data files. The project data file records are copied one record at a time to ensure that the data are transferred correctly. When this routine function is completed, the project data file on the designated disk drive is erased (KILLed), and control is returned to the Camouflage Program main routine. The source codes for the final routine function are provided below:

```
200 REM
210 REM COPY CONTENTS OF DAT FILE TO FLOPPY BEFORE RETURNING
220 REM COPY, CLOSE AND RETURN
230 FL$="P"+MID$(PN$,1,6)+MID$(RN$,1,1)+".DAT"
240 B$(1)=STR$(NXRC-1)+"  Records Copied to   A:"+FL$
250 IF LEN(B$(1))<78 THEN B$(1)=B$(1)+" ":GOTO 250
260 COLOR 31,1,1:LOCATE 24,2:PRINT B$(1);
270 OPEN "A:"+FL$ AS #2 LEN=128:FIELD #2, 128 AS BA$
280 GET #1,1:NXRC=CVS(MID$(AA$,1,4))
290 LSET BA$=AA$:PUT #2,1
300 FOR I=2 TO NXRC-1:GET #1,I:LSET BA$=AA$:PUT #2,I:NEXT I
310 COLOR 15,1,1:CLOSE:KILL FL$
320 REM
330 END SUB
```

Each of the input/edit routines called by the CAMINP routine CALL a second routine to display the contents of each screen. These screen routines are referenced by name in the descriptive text above, and the source codes for each screen routine are provided below. These screen routines are stored in a program module **CAMOSCRN** and are linked to the **CAMINP** routine.

```
10 REM  CAMOUFLAGE SYSTEM INPUT SCREENS       CAMOSCRN.BAS
20 REM  LATEST REVISIONS     01/16/91
30 REM
40 SUB SITESCRN(ESN$,RV$)
50 CLS
60 LOCATE 1,27:PRINT "FIXED FACILITY SUPPORT SYSTEMS"
65 LOCATE 1,62:PRINT "PROJECT # ";:PRINT USING "\    \";ESN$;
70 LOCATE 2,33:PRINT "Site Specific Data"
75 LOCATE 2,61:PRINT "REVISION # ";:PRINT USING "\\";RV$
80 LOCATE 4,10:PRINT "SOILS INVESTIGATION and FOUNDATION DATA"
90 LOCATE 6,20:PRINT "cohesion (psf):"
100 LOCATE 7,20:PRINT "unit weight (pcf):"
110 LOCATE 8,20:PRINT "angle of internal friction (degrees):"
```

```
120 LOCATE 9,20:PRINT "diameter of boring for pole socket
    (inches):"
130 LOCATE 11,10:PRINT "CLIMATE DATA"
140 LOCATE 13,20:PRINT "average daily rainfall (inches):"
150 LOCATE 14,20:PRINT "average daily temperature (degs F):"
160 LOCATE 15,20:PRINT "wind speeds (mph):"
170 LOCATE 16,25:PRINT "sustained average (mph):"
180 LOCATE 17,25:PRINT "gust speeds (mph):"
190 LOCATE 18,20:PRINT "potential ice or snow load (psf):"
200 LOCATE 21,10
205 PRINT "GENERAL TERRAIN AND FOLIAGE CLASSIFICATION:"
210 END SUB

220 SUB GEOMSCRN(ESN$,RV$)
230 CLS
240 LOCATE 1,27:PRINT "FIXED FACILITY SUPPORT SYSTEMS"
245 LOCATE 1,62:PRINT "PROJECT # ";:PRINT USING "\   \";ESN$;
250 LOCATE 2,33:PRINT "Structure Geometry"
255 LOCATE 2,61:PRINT "REVISION # ";:PRINT USING "\\";RV$
260 LOCATE 4,10:PRINT "PLAN VIEW DATA"
270 LOCATE 6,20:PRINT "length of structure (feet):"
280 LOCATE 7,25:PRINT "bay spacings (measured along the
    structure length)"
300 LOCATE 9,30:PRINT "@         @         @         @         @"
320 LOCATE 11,30:PRINT "@         @         @         @         @"
340 LOCATE 14,20:PRINT "width of structure (feet):"
345 LOCATE 15,20:PRINT "minimum pole spacing (feet):"
350 LOCATE 17,10:PRINT "ELEVATION DATA"
360 LOCATE 19,20:PRINT "exterior pole height (above ground feet):"
370 LOCATE 20,20:PRINT "exterior poles guyed (Y/N):"
380 LOCATE 21,20:PRINT "allowable netting sag (feet):"
385 LOCATE 22,20:PRINT "allowable displacement error (inches):"
390 END SUB
```

```
400 SUB POLSCRN(ESN$,RV$)
410 CLS
420 LOCATE 1,27:PRINT "FIXED FACILITY SUPPORT SYSTEMS"
425 LOCATE 1,62:PRINT "PROJECT # ";:PRINT USING "\   \";ESN$;
430 LOCATE 2,27:PRINT "Support Member Material Screen"
435 LOCATE 2,61:PRINT "REVISION # ";:PRINT USING "\\";RV$
440 LOCATE 4,3
445 PRINT "INDICATE DESIRED SUPPORT MEMBER TO USE (Y-Yes  N-No)"
450 LOCATE 6,21:PRINT "ALLOW    MODULUS"
460 LOCATE 6,38:PRINT "X-SECTION  MOMENT   UNIT"
470 LOCATE 7,12:PRINT "LENGTH  STRENGTH  ELASTIC"
480 LOCATE 7,40:PRINT "AREA    INERTIA   PRICE"
490 LOCATE 8,5:PRINT "MARK    (FT)    (KSI)    (10**6)"
500 LOCATE 8,39:PRINT "(SQIN)   (IN**4)  ($/FT)    DESCRIPTION"
510 REM
520 END SUB

530 SUB NETSCRN(ESN$,RV$)
540 CLS
550 LOCATE 1,27:PRINT "FIXED FACILITY SUPPORT SYSTEMS"
555 LOCATE 1,62:PRINT "PROJECT # ";:PRINT USING "\   \";ESN$;
560 LOCATE 2,31:PRINT "Netting Material Screen"
565 LOCATE 2,61:PRINT "REVISION # ";:PRINT USING "\\";RV$
570 LOCATE 4,3
575 PRINT "INDICATE DESIRED NETTING TYPE TO USE  (Y-Yes  N-No)"
580 LOCATE 6,12:PRINT " UNIT    NETTING   ALLOW"
590 LOCATE 6,40:PRINT "UNIT     DRAG      UNIT"
600 LOCATE 7,12:PRINT " SIZE     THICK  STRENGTH"
610 LOCATE 7,39:PRINT "WEIGHT   COEFFIC   PRICE"
620 LOCATE 8,5:PRINT "MARK   (SQFT)     (IN)    (KSI)"
630 LOCATE 8,38:PRINT "(#/SQFT)  (UNITS) ($/SQFT)   DESCRIPTION"
640 REM
650 END SUB

660 SUB TENSCRN(ESN$,RV$)
670 CLS
680 LOCATE 1,27:PRINT "FIXED FACILITY SUPPORT SYSTEMS"
685 LOCATE 1,62:PRINT "PROJECT # ";:PRINT USING "\   \";ESN$;
690 LOCATE 2,27:PRINT "Tension Member Material Screen"
695 LOCATE 2,61:PRINT "REVISION # ";:PRINT USING "\\";RV$
700 LOCATE 4,3
705 PRINT "INDICATE DESIRED TENSION MEMBER TO USE  (Y-Yes  N-No)"
710 LOCATE 6,13:PRINT "ALLOW   MODULUS  THERMAL"
720 LOCATE 6,37:PRINT " X-SECTION  UNIT      UNIT"
730 LOCATE 7,11:PRINT "STRENGTH  ELASTIC  COEFFIC"
740 LOCATE 7,40:PRINT "AREA    WEIGHT    PRICE"
750 LOCATE 8,5:PRINT "MARK   (KIPS)   (10**6) (10**-6)"
760 LOCATE 8,39:PRINT "(SQIN)   (#/FT)   ($/FT)    DESCRIPTION"
770 REM
780 END SUB
```

```
790 SUB ANCSCRN(ESN$,RV$)
800 CLS
810 LOCATE 1,27:PRINT "FIXED FACILITY SUPPORT SYSTEMS"
815 LOCATE 1,62:PRINT "PROJECT # ";:PRINT USING "\   \";ESN$;
820 LOCATE 2,31:PRINT "Anchor Material Screen"
825 LOCATE 2,61:PRINT "REVISION # ";:PRINT USING "\\";RV$
830 LOCATE 4,3
835 PRINT "INDICATE DESIRED ANCHOR TO BE USED  (Y-Yes  N-No)"
840 LOCATE 6,21:PRINT "ALLOW    MODULUS"
850 LOCATE 6,38:PRINT "  ROD       ANCHOR   UNIT"
860 LOCATE 7,12:PRINT "LENGTH  STRENGTH  ELASTIC"
870 LOCATE 7,38:PRINT "DIAMETER  DIAMETER  PRICE"
880 LOCATE 8,5:PRINT "MARK   (INCH)   (KIPS)   (10**6)"
890 LOCATE 8,39:PRINT "(INCH)    (INCH)  ($/EA)    DESCRIPTION"
900 REM
910 END SUB
```

6 DESIGN Routine

INTRODUCTION. This chapter presents a step-by-step procedure to design tent-like tension structures. The structures in question are cable networks supported by slender columns or poles cantilevered in the ground as shown in Figure ȷ. The exterior poles may be designed with or without guys. This network of cables and poles is then covered by fabric or netting to produce the tent-like shelter. The covering will be supported by the cable network but not attached in any way.

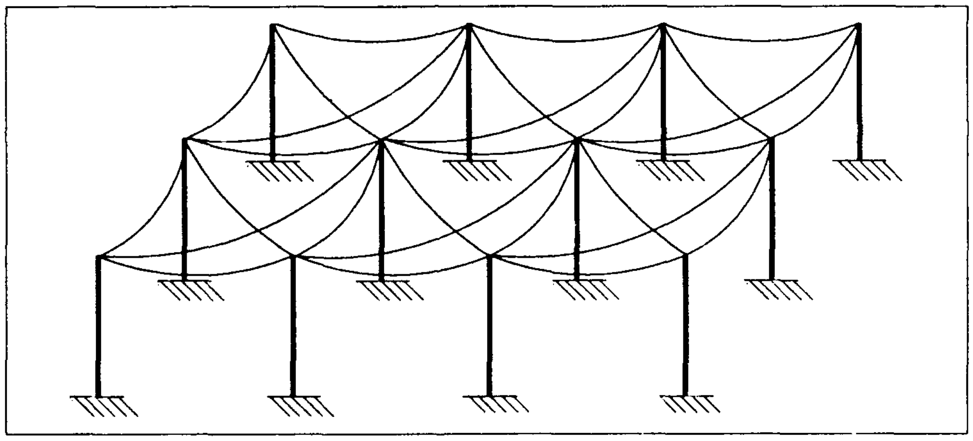

Figure 3. A typical tension structure

The basis of this chapter is the theory for tension structures. The main load-carrying members in tension structures transfer the applied loading to the supports by tensile stresses only. No compression or flexure occurs in these primary members. Cables or ropes will be the main load-carrying members used herein, and the columns or poles will be designated as the support structure.

The applications for these tension structures are mainly for temporary usage, although a well-constructed structure may be used as a permanent fixture. The tent-like structure is well suited as a temporary shelter for people, equipment, or job sites. These structures provide excellent shelter from sun, wind, and rain. Another use for these structures is the concealment of military equipment. These structures can be used to camouflage military equipment in any terrain. The simplicity of these structures makes erection quick and easy which is important in the process of concealment of militarily vital equipment and personnel. Not only do the tent structures conceal an area well, but they also provide excellent shelter from the elements.

Tension structures are well suited for these applications because of their efficiency in carrying distributed loads. As shelters or concealment structures, the loads carried are all distributed loads. The primary loads carried in these structures are caused by the weight of the cables and fabric (including the wet weight of the fabric due to rain), and snow and wind loads.

Other advantages of tension structures are given by Leonard (1988):

(1) Tension structures are lightweight, collapsible, easy to transport and erect, and relocatable.

(2) Tension structures may be prefabricated.

(3) Tension structures have low installation costs.

(4) Tension structures handle distributed environmental loads well through direct stress without bending.

(5) Tension structures are load adaptive. They change geometry to carry loads and changes in loads more efficiently.

Leonard (1988) states that the behavior of a tension structure can be divided into three phases in its life as a load-carrying body. The first phase is the deployment phase and is simply the unfolding of the system from its collapsed state in transport. During this phase the structure is stress-free, but care must be used to ensure that no kinks occur in the cables. The second phase is the prestressing phase in which the cables are attached to the support structure and possibly tightened to a preselected point on the cable or a preselected tension in the cable. The loads being carried in this phase are the weight of the cable and the prestressing forces. The final phase is the in-service phase when the prestressed system is subjected to all of the other dead and live loads over its lifetime.

The deployment phase involves no stresses since the structure has only been unfolded and not yet erected. This phase will not be included in the design procedure.

The prestressing phase is the point at which the structure is first erected and the cables are tightened in a specified manner to obtain the general shape of the structure. As the cables are tightened, the structure changes its geometry. Most of the geometry changes in this phase are simply changes in sag in the cables. Very little strain occurs in the cables at this point because of light external loads. The prestressing of the cables causes changes in sag and very small strains in the cable.

The in-service phase has the structure loaded with the larger loads of wind, rain, and snow. The changes in geometry that the structure exhibits at this time also involves changes in sag; but now with the higher loads, the cables begin to have larger strains. The in-service phase has the highest strains as the prestressing phase has the largest changes in sag.

PROCEDURE. The difficulty in designing or analyzing tension structures lies in the nonlinear behavior of the cables. With normal structural members, the deflection of the member is proportional to the load. When a cable is allowed to sag, this proportionality between load and displacement does not hold true. For a cable, the load increment causing a second inch of displacement may be considerably higher than the load causing the first inch of deflection. The nonlinearity is caused by the behavior of sag in a cable. As tension is applied to the cable, sag is reduced and strain occurs. When sag is large, an increase in tension causes very little strain in the cable. Most of the force is used in reducing the sag in the cable. As sag becomes smaller, increases in tension begin to cause more strain increase and less sag decrease. As the sag becomes small, more force is needed to displace the ends, as the effort is now in stretching the cable rather than in pulling out the sag. Reducing sag takes a small force compared to straining the cable.

The nonlinearity of the system does not allow a simple solution by matrix methods. Iterative solutions must be incorporated into the design procedure to allow the solution of the nonlinear system. For a structure of any size, hand calculations are incomprehensible; computer solution is the only efficient means of design. The procedure used herein is an iterative stiffness matrix solution for which each iteration is a linear solution. The iterations converge to a final solution as described in the following pages. The nonlinearity does not allow superposition of loadings, so all loads are applied at once at the beginning of the procedure to create the case of maximum expected tension in each cable. The procedure does however design for wind load from each of three directions (the fourth direction is always symmetric to another direction as will be shown later).

MODELING AND ASSUMPTIONS. The tension structure is modeled as a two dimensional grid in the plane of the plan view of the structure. The top of each pole is a node, and the nodes are numbered in the manner shown in Figure 4. Each node has two degrees of freedom: one each in the X- and Y-directions shown in Figure 4. The degrees of freedom are numbered in a manner such that the degree of freedom in the X-direction is one less than twice the node number, and the degree of freedom in the Y-direction is twice the node number. Although the grid shows the poles only as node points, the bending stiffness of the poles are included in the model. Displacements in the vertical direction are neglected.

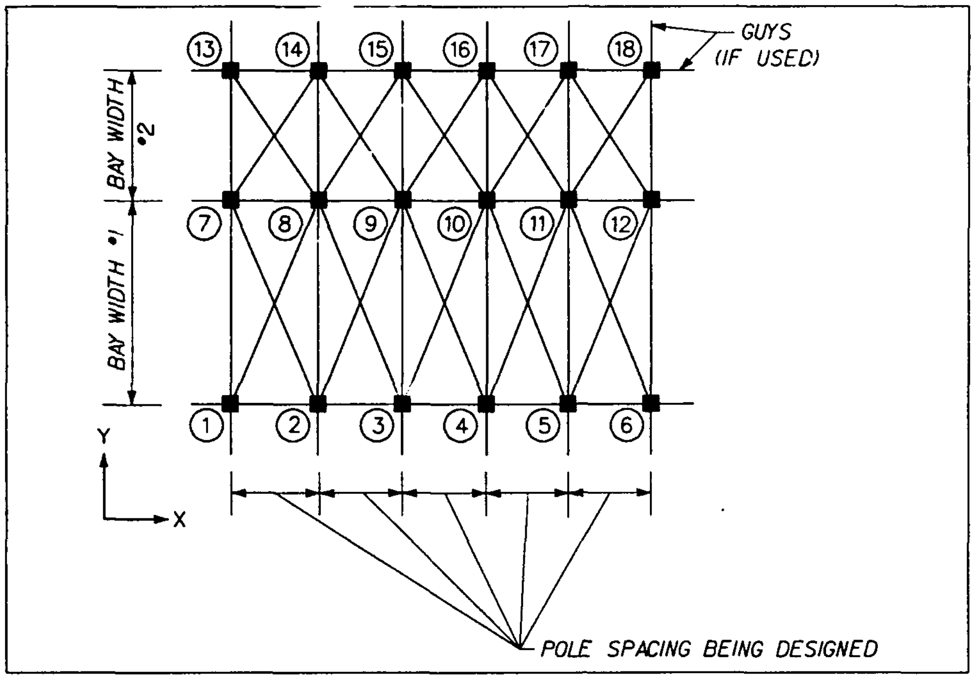

Figure 4. Plan view of a typical tension structure

All structures are rectangular and are subdivided into smaller rectangles. Each smaller rectangle has a cable forming each side and a cable connecting each pair of diagonally opposite nodes.

The stiffness for a joint is calculated as a stiffness in each of the X- and Y-directions. The stiffness for a joint is the stiffness of the pole added to the linearized stiffness of the cables which are put in tension by the collective force on the node.

The first assumption made in this design is that of small deflection theory. Simple cable nests allow large deflections to occur, but the structure being designed has the restraint of the columns. Large deflections would cause column failure. To have a functioning system, small deflections at the nodes is a reasonable assumption.

The second assumption is that sag ratios are small. This assumption must be made in order for the equations used in this procedure to be applicable. This assumption is reasonable because cables with large sag ratios are not efficient in transmitting loads.

DESIGN INPUT. The given information for the design procedure must include the material properties and sizes, structure size, loads, and whether the structure is guyed or not. The material properties and sizes of the poles must include modulus of elasticity, moment of inertia, yield stress, height, cross-sectional area, cross-sectional width, and minimum distance between poles. The material properties and sizes of the cables must include modulus of elasticity, maximum tension strength, maximum sag, cross-sectional area, and linear weight. The structure size must include the number of bays in one direction and the size of those bays, and the total length in the other direction. The direction perpendicular to the bay widths is the direction being designed. The loads other than cable weight are the fabric wet weight, the snow load, and the wind load. The angle between the fabric and the ground and the fabric's drag coefficient must also be given. A choice of guyed or nonguyed exterior poles is given. And finally the factor of safety to be used for the structure must be given. More detail on design input is given in Appendix B.

BASIC DESIGN MAP. Once the above information is known, the design process begins. The points of design are the number of equally-spaced poles in the direction perpendicular to the specified bay widths, and the initial prestressing for each cable.

The design begins by calculating the maximum and minimum number of equally spaced poles in the design direction. The maximum number of poles is defined by the user's input of the minimum distance allowed between columns. The minimum number of poles is always two (in the case that the design direction will have poles only at the extreme corners of the structure). The design continues until the minimum number of poles without a column or cable failure is found. The minimum number of poles defines the maximum column spacing and therefore the most efficient structure. The possibility exists that the structure will fail even at the minimum spacing specified by the user, so the procedure has mechanisms to check for this case. This case is checked first so that no more time is used in the design if the design is impossible because of column or cable failure for the given information.

The procedure begins by using the maximum number of poles as the first trial. If this case fails, notification is given and the procedure ends. If the first case passes the failure criteria, then a new trial number of poles is used. This next number of poles is the midpoint of the maximum and minimum number of poles. If a trial number of poles passes the failure criteria, then this number becomes the new maximum number of poles so that no more poles than necessary are used. If a trial number of poles fails the failure criteria, then this number becomes the new minimum number of poles since it is now known that more poles are needed. To choose the next trial number of poles, the midpoint of the new maximum and minimum is used. This process continues until convergence to the minimum number of poles to pass the failure criteria.

This basic plan is followed to find the minimum number of poles to satisfy the given information without failure of column or cable. The specifics in the design can be broken into six steps: (1) calculating basic cable geometry, (2) calculating applied loads on cables, (3) calculating proper unstressed length of cable, (4) preparing direct stiffness method, (5) solving equations, and (6) interpreting final data.

BASIC CABLE GEOMETRY. Each cable in a structure trial has a number, a beginning node, and an ending node. The specifics for obtaining this information is given in Appendix B.

Four ratios are calculated for each cable. "FXFL" is the force in the component X-direction caused by a unit force along the cable's length. "FYFL" is the force in the component Y-direction caused by a unit force along the cable's length. "DLDX" is the deflection along the length of the cable caused by a unit deflection in the component X-direction. "DLDY" is the deflection along the length of the cable caused by a unit deflection in the component Y-direction.

Each cable is also classified according to its location and use in the structure. Nine classifications exist specifying the use of the cable, the location of the cable, and the direction of the cable. A cable is used either as part of the roof nest or as a guy. The location of the cable is either interior or exterior. The direction of the cable is either parallel to the bay widths, parallel to the designed pole spacings, or diagonal. The specifics of these classifications can also be found in Appendix B.

APPLIED FORCES ON CABLES. Applied loads on the cables come from four sources: (1) the weight of the cable, (2) the weight of the netting or covering material and the moisture that it can hold, (3) the weight of accumulated snow, and (4) the force of the wind. For the purposes of this design, these forces are reduced to a uniformly distributed load over the length of the cable. The uniformly distributed loads must at this time be estimated because the true length of the cable is not known yet. The horizontal projection of each cable is defined by the given bay widths and the trial pole spacing in the design direction, but the true length has to be calculated based on the loads and the amount of sag allowed.

A cell in the structure will be defined as a rectangular region with dimensions of bay width by trial pole spacing. As shown in Figure 5, each cell is divided into four triangles by the two diagonal cables. To estimate the uniformly distributed loads on the cables, each of these triangles will be divided into three more triangles by connecting the vertices of a triangle to the centroid of that triangle as shown by the dashed lines in Figure 6. Each cell is now divided into twelve triangles. Each triangle has exactly one side in contact with a cable. The load carried by each triangle is considered to be carried by the cable which makes up one of its sides. The load on each cable is the uniformly distributed load, which is equal in total force to the loads carried by the triangular areas associated with each cable. Equations and other specifics for calculating these loads are given in Appendix B.

UNSTRESSED LENGTH OF CABLE. The unstressed length of each cable can be calculated if the uniformly distributed load along the length of the cable is known and the sag in the cable is set to eighty percent of the maximum allowed sag. This unstressed length of cable will define the initial prestressing stage of the structure. The unstressed length

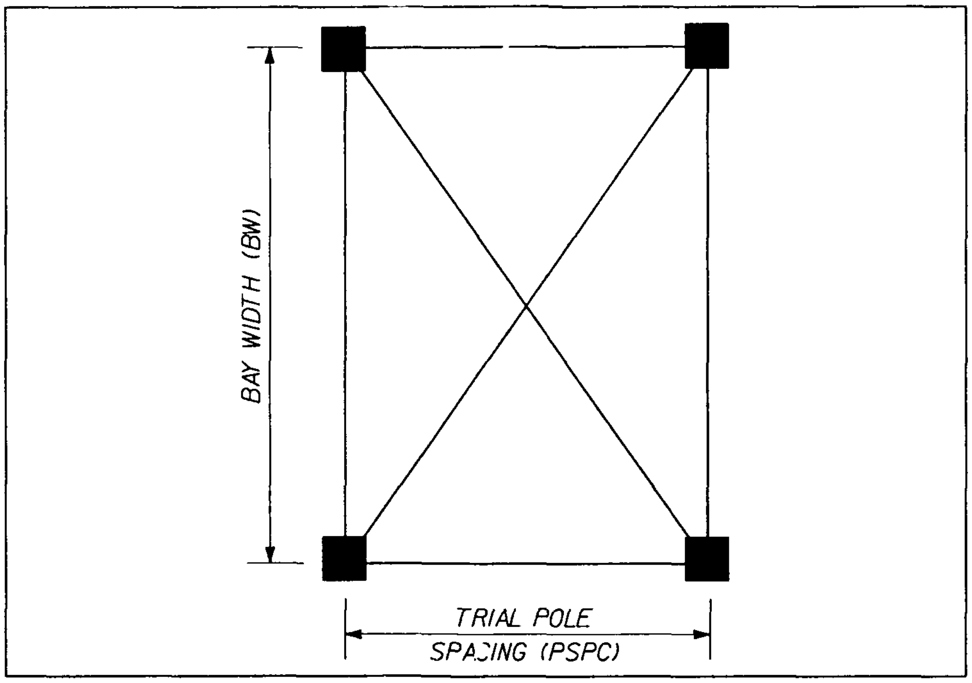

Figure 5. Single cell plan view

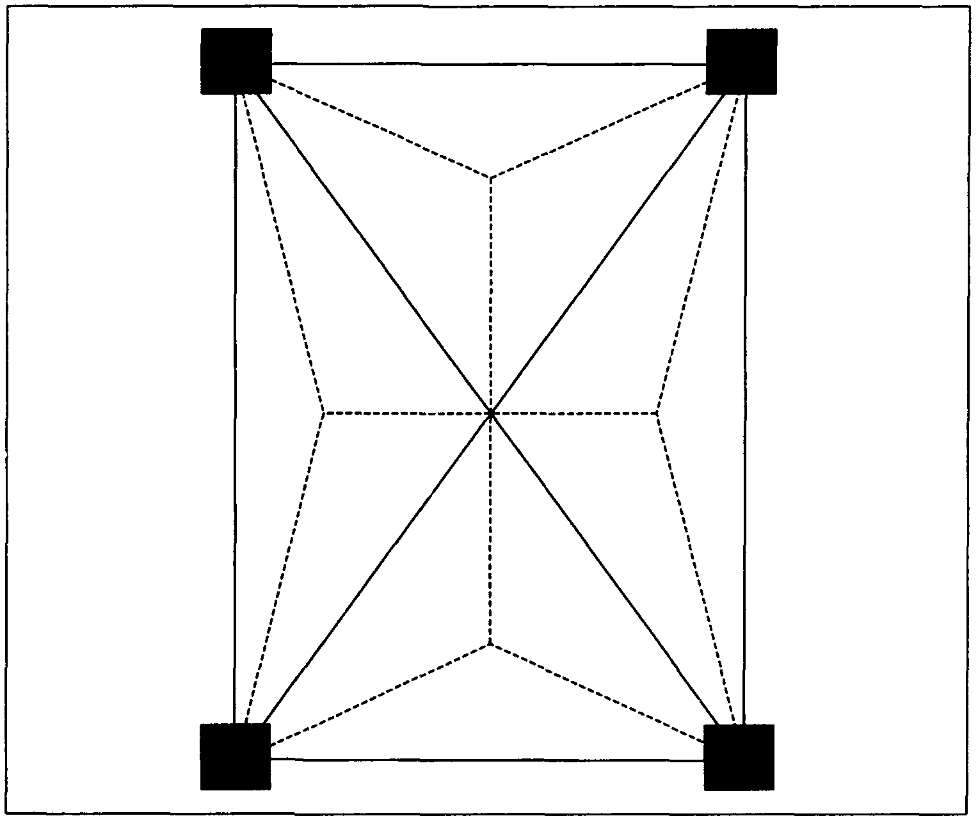

Figure 6. Single cell load divisions

is the length the cable must be initially before any kind of stress is added to the cable. The cable must be marked, hung in place, then tightened to the markings so that the cable has the correct initial prestressing.

For each cable, the sag is initially set at eighty percent of the maximum sag allowed. Calculating the sag ratio for each cable we have:

$$f_0 = \frac{0.8 * SMAX}{L_0} \tag{1}$$

From Leonard (1988) we have the following:

$$f = \frac{q * L}{8 * H_0} \tag{2}$$

$$\beta = \frac{q * L}{2 * H_0} \tag{3}$$

$$\gamma = \sinh^{-1}\left(\frac{\beta}{\sinh\beta} * \tan\theta\right) \tag{4}$$

Rearranging Equation 2 and substituting initial conditions into all three we obtain:

$$H_0 = \frac{q * L_0}{8 * f_0} \tag{5}$$

$$\beta_0 = \frac{q * L_0}{2 * H_0} \tag{6}$$

$$\gamma_0 = \sinh^{-1}\left(\frac{\beta_0}{\sinh\beta_0} * \tan\theta\right) \tag{7}$$

From Leonard (1988) we find:

$$\frac{S_0}{L_0} = \frac{V_0 - V_1}{qL_o} - \frac{qL_0}{2AE}\left[\frac{H_0}{qL_0} + \frac{V_0T_0 - V_1T_1}{(qL_0)^2}\right] \tag{8}$$

$$V = H_0\sinh\left[\gamma + \beta\left(1 - \frac{2x}{L}\right)\right] \tag{9}$$

$$T = H_0\cosh\left[\gamma + \beta\left(1 - \frac{2x}{L}\right)\right] \tag{10}$$

Recognizing the fact that V_0 and T_0 are at $x=0$, and V_1 and T_1 are at $x=L_0$, inserting this information into Equation 8, and solving for the unstressed length of cable S_0, we find

$$S_0 = \frac{H_0}{q}\left[\sinh\left(\gamma_0 + \beta_0\right) - \sinh\left(\gamma_0 - \beta_0\right)\right]$$

$$- \frac{H_0 L_0}{2AE}\left\{1 + \frac{H_0\,[\sinh(\gamma_0 + \beta_0)\cosh(\gamma_0 + \beta_0) - \sinh(\gamma_0 - \beta_0)\cosh(\gamma_0 - \beta_0)]}{qL_0}\right\} \qquad (11)$$

Knowing the unstressed length of the cable aids in the erection of the structure. It is much quicker and efficient to be able to tighten a cable to a premarked point than to try to tighten the cable to a certain force. If tightening the cables to a certain prestressing force is tried, it is found that as one cable is tightened, it also prestresses the other cables which have already been prestressed to the desired amount. This action would cause over stressing of many cables. Calculating a length of cable which can be tightened to a certain mark is a simple and quick task. The unstressed length of each cable is stored in the program in the tenth column of the "ZZ" array. When the program has finished, the unstressed length of each cable should be printed in order to correctly erect the structure in the prestressing phase. It is seen that cables of the same classification and in the same bay have the same unstressed length.

SETTING UP DIRECT STIFFNESS METHOD. The next step is to set up the iterative method for solving the nonlinear problem. The method used for each iteration is a simple direct stiffness method for the equation:

$$\{F\} = [K] * \{U\} \qquad (12)$$

where *[F]* is the array of X and Y forces at each node, *[K]* is the matrix of X and Y stiffnesses at each node, and *{U}* is the array of X and Y displacements at each node for which the equations must be solved.

The initial forces at the nodes are calculated as if the structure begins with the poles perfectly rigid. Using Equation 5, H_0 is calculated for each cable and stored in positions in array "ZZ". H_0 is the tensile horizontal force in line with the horizontal projection of the cable. L_0 is the original length of the horizontal projection of the cable. Multiplying the H_0 force by the coefficients "FXFL" and "FYFL", the horizontal force along the length of the horizontal projection is transformed into the X- and Y-component forces to be used in the *{F}* array. Continuously adding the forces of each cable to the nodes to which they attach, keeping track of signs (positive is to the left for the X-direction, and upwards for the Y-direction), the end result is the resultant force on the top of the pole.

The initial stiffnesses of the structure depend on the resultant forces in the {F} array. In calculating the stiffness at a joint, only the stiffnesses of the cables which are stretched by the resultant force are included in the stiffness for that joint. The stiffness at a joint is composed of the stiffness of the pole (which is a constant at every node) and the stiffnesses of the cables which are put in tension by the resultant force on the node. The stiffness for each cable is taken as the force needed to stretch the cable a unit of tensile deflection.

From Leonard (1988) we have:

$$H = H_0 * \left(1 + K_0 * \frac{\Delta L}{L_0}\right) \tag{13}$$

$$K_0 = 1 + \frac{K_{01} - \dfrac{S_0}{L_0 + \Delta L}}{K_{02}} \tag{14}$$

$$K_{01} = \tan^2\theta + \frac{q * L_0}{8 * A * E * f_o} * \left(1 + \tan^2\theta + \frac{16}{3} f_0^2\right) \tag{15}$$

$$K_{02} = \frac{S_0}{L_0} - 1 - \frac{1}{2}\tan^2\theta - 8 * f_0^2 + \frac{3 * q * l_0}{16 * A * E * f_0} \tag{16}$$

Looking at these equations, we can see that K01 and K02 are constants. These values are calculated for each cable and stored in the "ZZ" array. Equation 14 becomes the stiffness equation if a unit deflection is input in the equation for the delta term. H becomes the force needed to pull the ends of the cable a unit distance away from each other. Using the "FXFL" and "FYFL" terms to break the stiffness into X- and Y-component stiffnesses, the stiffness matrix is created.

SOLUTION OF EQUATIONS. Knowing the force vector and the stiffness matrix, the equations can be solved for the deflection vector. The first deflection vector is not the final deflections for this trial pole spacing. Due to the nonlinear behavior of cables and the fact that cables do not exert force in compression, these initial deflections are not the final deflections. Using the deflections which have been calculated, a new force vector and stiffness matrix is calculated. The new force matrix can be calculated by using Equation 14 with the calculated deflections from the iteration before to arrive at the new forces. The stiffness matrix is again calculated as the stiffness of the pole and the stiffnesses of the cables which are stretched by the new force matrix. Again Equation 14 is used to calculate the stiffness except that the delta term is now the deflection from the previous iteration plus a unit deflection. A new displacement vector is calculated from the new force vector and stiffness matrix. The new displacement vector is compared to the old displacement vector, and if every corresponding pair of displacements agree with each other to a specified accuracy, then the average of the two sets of displacements is taken as the final displacements. If the two sets of displacements do not correspond to the desired accuracy, then the two arrays are averaged to find a new initial guess, and the procedure begins again by calculating a new force vector and stiffness matrix based on the new set of averaged displacements.

For each trial pole spacing, the procedure is carried through to find the final displacements for three load cases. The first load case has all applied loads and the wind load from the bottom of the plan view. The second load case has all applied loads and the wind load from the left side of the plan view. The third load case has all applied loads and the wind load from the top of the plan view. The wind load from the right of the plan view is not calculated since it is symmetric to the wind load from the left case. Wind loads are covered in Appendix B.

When the final deflections for a load case and trial pole spacing is found, the final forces in all cables and final stresses in all poles are then calculated and compared to the allowable forces or stresses for each element. The forces in each cable are compared to the given maximum tensile force for which the cable can hold reduced by the given factor of safety. The maximum stress in each pole occurs at the base. Based on the vertical force each pole supports and the moment created by the deflections at the top of the pole, the stress can be calculated and compared to allowable stresses for columns subjected to bending. The equations used for these purposes are found in Appendix B.

FINAL OUTPUT. The output needed in order to erect the designed structure is the designed pole spacing which is stored as "PSPC" and the unstressed length of each cable. The unstressed length is the fully-supported unstressed length which should be marked on the cable in order that the cable can be put in place and "stretched to" the mark.

References.

Leonard, W. M. (1988). *Tension structures.* McGraw-Hill Book Company, New York, NY.

7 FOOTING Routine

The FOOTING routine is designed to calculate the required depth of the foundations for the individual support members. The routine is CALLed by the MAIN program at the completion of the design of the specified structural system (Chapter 1 Main Program). The arguments to the routine are as follows:

(1) Unit weight of the soil, GAMMA, pounds/ft^3

(2) Soil cohesion, COH, pounds/ft^2

(3) Angle of internal friction of the soil, PHI, degrees

(4) Diameter of the foundation, DIA, inches

(5) Height of support member, HT, feet

(6) Maximum moment in support member, M, foot-pounds

The FOOTING routine iteratively determines the depth of the foundation required to resist the maximum moment and maximum lateral force transmitted from the support member. The development of the formulations are from Czerniak (1957). An initial depth of 0.25 ft is selected, and the moment and lateral force on the foundation are calculated. The depth is incremented by 0.25 ft until the calculated moment and lateral force are less than the allowable values. The final output of the routine is written to the scratch sequential data file named FOOTING.DAT. The final output values are as follows:

(1) Derived depth to prevent overturning, feet

(2) Applied moment for derived depth, foot-pounds

(3) Applied force for derived depth, pounds

(4) Allowable moment, foot-pounds

(5) Allowable force, pounds

The formulations required to calculate the moment and lateral force as a function of foundation depth are as follows:

$$P_h = k\gamma h \tan^2\left(\frac{\theta}{2} + 45\right) + 2c \tan\left(\frac{\theta}{2} + 45\right) \tag{1}$$

where

P_h	= horizontal pressure against vertical plane, psf
k	= efficiency factor (assumed 2.0)
γ	= unit weight of the soil, pcf
h	= foundation depth, feet
θ	= angle of internal friction, degrees
c	= soil cohesion, psf

$$p = \frac{\left(\frac{a}{h}\right)^2 P_a}{4\left(1 - \frac{a}{h}\right)} \tag{2}$$

where

p	= earth pressure against pile at a distance of a/2 from the resisting surface, psi
a	= distance from resisting surface to pivot point, 0.70h, feet
P_a	= earth pressure against pile at a distance h, $(2/3)p_h$, psf

$$T = \frac{2}{3} pa - \frac{2}{3} \frac{ph^2 (3a - 2h)}{a^2} \tag{3}$$

where T = total pressure on the back of the pile, pounds per foot of diameter

$$M_0 = -ph^3 \frac{(4a - 3h)}{3a^2} \tag{4}$$

where M_o = moment per foot of pile diameter, applied at the resisting surface, foot-pounds per foot of diameter

The source codes for the FOOTING routine are provided below:

```
3910 SUB FOOTING(GAMMA,COH,PHI,DIA,HT,M)
3920 REM NEW PROGRAM TO CALCULATE DEPTH OF FOOTING
3950 REM
3960 REM **** A=DISTANCE TO PIVOT POINT
3965 REM      P=EARTH PRES. ON PILE AT DIST. A/2:
3970 REM      FO=LATERAL FORCE PER FOOT OF PILE DIA. = F/DIA*12
3980 REM      MOF=MOMENT PER FOOT OF PILE DIA. = M/DIA*12
3990 REM      PH=PASSIVE EARTH PRESSURE: PA=(2/3)*PH
4000 REM
4010 F=M/HT:FO=F/DIA*12:MOF=M/DIA*12
4020 K=2:NPHI=(TAN((45+PHI/2)*(3.141592654#/180)))^2
4030 H=0.25
4040 A=.7*H:PH=K*GAMMA*H*NPHI+2*COH*SQR(NPHI)
4050 PA=(2/3)*PH:P=(A/H)^2*PA/(4*(1-(A/H)))
4060 T=2/3*P*A-2/3*(P*H^2/A^2)*(3*A-2*H)
4070 MO=-P*H^3*(4*A-3*H)/(3*A^2)
4080 IF T>FO AND MO>MOF THEN 4110
4100 H=H+0.25:GOTO 4040
4105 REM
4110 OPEN "FOOTING.DAT" FOR OUTPUT AS #1
4120 A$="LENGTH TO PREVENT OVERTURNING        ######.### FEET"
4130 PRINT #1,USING A$;H
4140 A$="MOMENT APPLIED                      ######.### FT-POUNDS"
4150 PRINT #1,USING A$;MOF*DIA/12
4160 A$="ALLOWABLE MOMENT                    ######.### FT-POUNDS"
4170 PRINT #1,USING A$;MO*DIA/12
4180 A$="LATERAL FORCE APPLIED               ######.### POUNDS"
4190 PRINT #1,USING A$;FO*DIA/12
4200 A$="ALLOWABLE LATERAL FORCE             ######.### POUNDS"
4210 PRINT #1,USING A$;T*DIA/12
4220 CLOSE #1
4230 END SUB
```

Reference.

Czerniak, E. (1957 (Mar)). "Resistance to overturning of single, short piles," *ASCE, Journal of the Structural Division* (Paper 1188).

8 ANCDES Routine

The ANCDES routine is designed to calculate the capacity of the anchors selected for the structure. A maximum of five anchor types may be selected (see Chapter 5 CAMINP Routine). The routine is CALLed by the MAIN program at the completion of the footing/foundation design for the specified structural system (Chapter 1 Main Program). The arguments to the routine are as follows:

(1) Unit weight of the soil, GAMMA, pounds/ft^3

(2) Soil cohesion, COH, pounds/^{2}

(3) Angle of internal friction of the soil, PHI, degrees

(4) Maximum tension in the tension members, P, pounds

The remainder of the data required by the ANCDES routine are extracted from the labeled COMMON block ANCHORS which contains the data for up to five anchor types, as follows:

(1) Five-character name of the selected anchor, ANCMK$

(2) Length of the anchor rod, ANLEN, inches

(3) Allowable tensile strength of the rod, ANALL, kips

(4) Diameter of the anchor helix, ANDIA, inches

(5) Unit cost of the anchor, ACOST, \$/each

(6) 15-character description of the anchor, ANCDS$

A bearing capacity factor is calculated as a function of the angle of internal friction for the soil selected. If the soil is cohesionless (COH=0), the bearing capacity factor is set to zero. For each anchor whose rod tensile strength is greater than the tensile load applied, a bearing capacity is calculated as a function of the anchor helix area (AH), effective overburden pressure (Q), bearing capacity factor (NQ), and the soil cohesion

(COH). The formulations for these calculations are provided below. The program output is directed to the data file ANCHOR.DAT on the default disk drive. The output to this data file are as follows:

(1) Helix diameter, inches

(2) Assumed depth of the helix, inches

(3) Helix bearing capacity, pounds

(4) Maximum tensile load applied from the tension member, pounds

(5) Maximum tensile strength of the anchor rod, pounds

<u>Related Design Formulations</u>

$$NQ = 115.51541 - 23.77056\Phi + 1.71293\Phi^2$$

$$- 0.05528\varphi^3 + 0.000805\varphi^4 - 3.70883 \times 10.0^{-6}\,\varphi 5 \qquad (1)$$

$$Q = \gamma L \qquad (2)$$

$$A_h = \frac{\pi d^2}{4} \qquad (3)$$

$$Q_h = A_h\,(9.0c + Q \times NQ) \qquad (4)$$

where

NQ	= bearing capacity factor
Φ	= soil angle of friction, degrees
Q	= effective overburden pressure, pounds/ft^2
γ	= soil unit weight, pounds/ft^3
L	= assumed depth as effective length of the rod, feet
A_h	= projected helix area, ft^2
d	= helix diameter, feet
Q_h	= helix bearing capacity, pounds
c	= soil cohesion, pounds/ft^3

The source codes for the ANCDES routine are provided below:

```
4240 SUB ANCDES(GAMMA,COH,PHI,P)
4250 REM PROGRAM FOR ANCHOR DESIGN ON GUYED STUCTURES ONLY
4260 OPEN "ANCHOR.DAT" FOR OUTPUT AS #1
4270 PRINT #1, " Diameter     Depth     Capacity    Max Load
        Max Strength"
4280 PRINT #1, "  (INCH)      (INCH)      (LBS)       (LBS)
       (LBS)   "
4290 PRINT #1, "----------------------------------"
4300 A$=        "  ####.##  #####.##     ######.#   #######.#
                   #######.#  "
4310 NQ=115.51540878322#-23.77055843189#*PHI+1.71292587283#*PHI^2
4320 NQ=NQ-.05528105205#*PHI^3+.0008049511#*PHI^4-3.70883E
     -06*PHI^5
4330 FOR JZ=1 TO NANC
4335 DIA=ANDIA(JZ):D=ANLEN(JZ):RDSTR=ANALL(JZ)*1000!
4340 IF RDSTR >= P THEN
4350     AH=3.141592654#*(DIA/12)^2/4
4360     IF PHI=0 THEN NQ=0:GOTO 4380
4370     Q=GAMMA*D/12
4380     QH=AH*(9*COH+Q*NQ)
4390     PRINT #1, USING A$;DIA,D,QH,P,RDSTR
4400 ENDIF
4410 NEXT JZ:CLOSE #1
4420 END SUB
```

9 OUTPUT Routines

The Camouflage Program provides two major forms of program output immediately after the various designs of the structural system are completed. The first output form consists of a graphic display of a plan view of the structure with a printer option. The second output form consists of a complete record of the project input screens, structure design output, footing/foundation design output, anchoring design output, and a summary of the structural materials required. These data are written to a disk file, and through the **UTILITY** routine functions, may be reviewed on the screen, copied to floppy disk, dumped to the printer, or erased. Each of these output forms will be described and the associated source codes provided.

Graphic/Printer Plan View. At the successful conclusion of the design routines, the DRAW program is initiated. The DRAW program is a stand-alone executable routine written in FORTRAN and is accessed by the Camouflage MAIN routine by the SHELL command. The source codes for the DRAW program are provided in Appendix C. The inputs required by the DRAW program are written to the sequential data file name CARRY.DAT in the MAIN routine before the SHELL is initiated. The input requirements of the DRAW program are as follows:

1. Port number of the desired display device,
2. Model number of the desired display device,
3. Maximum size of the display in the X dimension in inches,
4. Maximum size of the display in the Y dimension in inches, and
5. A single line of descriptive text to be displayed along the bottom boundary of the graphic display.

The first four data items are provided on the first sequential line of input in the CARRY.DAT data file. These data are established when the **DEVICE** utility program (see Chapter 3 Utility Routines) is executed. The initial display of the structural plan view will be provided on the monitor of the computer system. The port and model numbers correspond to the type of monitor specified during the execution of the DEVICE program. The maximum X and Y sizes of the display are assumed to be 9.0 inches and 7.0 inches, respectively. The descriptive text consists of the project name, revision number, and 28 characters of fixed text. The plan view of the final structural design is displayed on the monitor until any keystroke is made. The source codes illustrating the data preparation for the initial execution of the DRAW program are provided below:

```
3480 REM DESIGN COMPLETE - GRAPHIC PROGRAM OUTPUT
3490 OPEN "DEVICES" FOR INPUT AS #1:NLINE=1
3500 LINE INPUT #1, A$:CALL SCAN(A$,V(),NN,B$(),NW)
3510 IF B$(1)="PRINT" THEN PPORT=V(1):PMODEL=V(2):PSCALE=V(3)
3520 IF B$(1)="SCREEN" THEN SPORT=V(1):SMODEL=V(2):PSCALE=V(3)
3530 IF NLINE=1 THEN NLINE=2:GOTO 3500
3540 CLOSE #1
3550 REM
3560 REM INPUT DATA FOR DRAW PROGRAM
3570 OPEN "CARRY.DAT" FOR OUTPUT AS #1
3580 XPAGE=9!:YPAGE=7!:C$=" #### #### #####.### #####.###"
3590 PRINT #1, USING C$;SPORT,SMODEL,XPAGE,YPAGE
3600 D$=" PROJECT: \    \  REV: \\       CAMOUFLAGE DESIGN
     PLAN VIEW"
3610 PRINT #1, USING D$;ESN$,RV$:CLOSE #1
3620 SHELL "DRAW < CARRY.DAT"
```

The contents of the sequential data file DEVICES were established during the execution of the DEVICE program. The type of monitor and printer were defined and the desired size of the printer plot was specified. The data for the port and model numbers of the monitor are extracted from the DEVICES data file as SPORT and SMODEL. These data and the assumed page size of 9.0 inches by 7.0 inches are written to the sequential data file CARRY.DAT. The project name and revision number are written to CARRY.DAT on the second line of sequential output. The SHELL to the DRAW program specifying that the program input is provided by the data file CARRY.DAT will generate the first display of the structural plan view.

The second display of the structural plan view is an optional display to the specified printer device. The following prompt is provided for this optional display:

Plot to Printer (Y/N) ?

If a printer plot is not desired, a response of N will suspend the graphic output for the project. A response of Y (or y) will provide a printer plot of the plan view with a page size as specified from the execution of the DEVICE program. At the conclusion of the printer plot, the generation of the second output form of the program is initiated. The source codes for the printer plot generation are provided below:

```
3640 COLOR 15,1,1:CLS:LOCATE 24,2:INPUT "Plot to Printer
     (Y/N) ";A$
3650 IF A$="y" THEN A$="Y"
3660 IF A$="Y" THEN
3670       XPAGE=10!:IF PSCALE>1! THEN XPAGE=13!
3680       YPAGE=7.5:IF PSCALE>1! THEN YPAGE=10!
3690       OPEN "CARRY.DAT" FOR OUTPUT AS #1
3700       PRINT #1, USING C$;PPORT,PMODEL,XPAGE,YPAGE
3710       PRINT #1, USING D$;ESN$,RV$:CLOSE #1
3720       SHELL "DRAW < CARRY.DAT"
3730 ENDIF
3740 KILL "CARRY.DAT"
```

The sequential data file CARRY.DAT is rewritten for a printer plot using the port (PPORT) and model (PMODEL) numbers of the specified printer and a page size of either 8.5 by 11.0 inches or 11.0 by 14.0 inches. One-inch borders on all sides are provided. At the conclusion of the printer plot generation, the data file CARRY.DAT is erased (KILLed).

DROUTPUT Routine. The final output form of the Camouflage Program consists of the complete project input, results of the design of the structure, results of the footing/foundation design, results of the anchoring design (if the structure were guyed), and a summary of the structural materials required. Each of these output forms are generated by the DROUTPUT routine, which is called by the MAIN routine immediately after the graphic displays are completed. The arguments of the DROUTPUT routine are as follows:

1. Six-character project name, ESN$,
2. One-character revision number, RV$,
3. Designated disk drive for the output, DD$,
4. Program user number, DLNUM, and
5. Program version number, PGVER$.

The designated disk drive, program user number, and program version number are extracted from the random access data file named COMFIL. These data were provided during the execution of the **VALID** program (see Chapter 3 Utility Routines). If this data file does not exist, execution will continue, and the output will be directed to the default disk drive (DD$=blank) of the computer system. The program user and version numbers will both be zero. All of the output generated by the DROUTPUT routine will be written to a sequential data file on the designated disk drive. The name of the output data file is defined by the project name and revision number as follows:

Pxxxxxxy.PRT

where

P	File name prefix
xxxxxx	Six-character project name
y	One-character revision number
PRT	File extension name

Project Input. The first output form generated consists of each input screen for which data were provided. A minimum of six and a maximum of ten screens will be displayed on the monitor screen and while being written to the designated .PRT data file at the same time. The output screens are as follows:

1. Site Specific Data Screen,
2. Structure Geometry Screen,
3. Support Member Material Screens (maximum of two),
4. Netting Material Screens (maximum of two),
5. Tension Member Material Screens (maximum of two), and
6. Anchor Material Screens (maximum of two).

The screen displays are provided in the DATADUMP routine, and the data file generation of the screens is provided in the PRINTFILE routine. The DATADUMP routine is CALLed from the DROUTPUT routine, and the PRINTFILE routine is CALLed from the DATADUMP routine. The arguments of the DATADUMP routine are the project name, ESN$, the revision number, RV$, and the designated output disk drive, DD$. The arguments of the PRINTFILE routine are the desired function number, NFUN, and the device number of the output data file, NFILE. Each of these routines will be described and source codes provided.

DATADUMP Routine. The DATADUMP routine successively reads the input screen data from the project data file, and if appropriate, displays the designated screen and its data on the monitor. Each screen is performed separately, and no user responses are required between screen displays. The project data file, data files for the field locations and field lengths, and the data file for the program output are OPENed before the screen displays are initiated. For each screen (NPR), the sequence of program steps is the same. The field locations and field lengths are extracted from the DATA.LOC and DATA.FLD data files and loaded into the LPTS() and FLD() arrays. If the beginning address of the screen data

(IREC(NPR)) is defined (not zero), the screen data are extracted from the project data file using the DSKRD routine and loaded into the VAR$() array. The appropriate screen routine is CALLed to display the desired screen, and the SCRNDUMP routine is CALLed to display the screen data. Following the screen display, the PRINTFILE routine is CALLed to append the screen display to the designated output file. A routine function (NFUN) of one specifies this desired function, and the output file device number (NFILE) is four. Two screens will be printed on each page of output. At the conclusion of the file output, a fixed border (====) is printed to subdivide the screen output on the page if the screen displayed is the first screen on the page. If the screen display is the second screen on the page, a form feed (CHR$(12)) is printed. The source codes for the DATADUMP routine are provided below:

```
3220 SUB DATADUMP(ESN$,RV$,DD$)
3230 REM
3240 COLOR 15,1,1:CLS
3260 FL$="P"+MID$(ESN$,1,6)+MID$(RV$,1,1)+".DAT"
3270 PL$="P"+MID$(ESN$,1,6)+MID$(RV$,1,1)+".PRT"
3280 OPEN DD$+FL$ AS #1 LEN=128:FIELD #1, 128 AS AA$
3290 OPEN DD$+"DATA.LOC" AS #2 LEN=4:FIELD #2, 4 AS DL$
3300 OPEN DD$+"DATA.FLD" AS #3 LEN=4:FIELD #3, 4 AS DF$
3310 OPEN PL$ FOR OUTPUT AS #4
```

The initial functions of the DATADUMP routine define the project data and the output data file names (FL$ and PL$), OPEN the project data file, OPEN the screen field locations and field lengths data files (DATA.LOC and DATA.FLD), and OPEN the output data file. The site specific data are contained on the first screen. These data are read from the project data file, and the site specific data screen and its data are written to the output data file. The source codes for this portion of the output are provided below:

```
3330 REM DUMP SITE SPECIFIC DATA ON FIRST PAGE
3340 NPR=1:CALL GETLOC(NPR,DUM())
3350 FOR I=1 TO NAD(NPR):LPTS(I)=DUM(I):NEXT I
3360 CALL GETFLD(NPR,DUM())
3365 FOR I=1 TO NAD(NPR):FLD(I)=DUM(I):NEXT I
3370 FREC=IREC(NPR):CALL DSKRD(FREC,NDUM,DUM(),DUM$())
3380 FOR I=1 TO NDUM:VAR$(I)=DUM$(I):NEXT I
3390 IST=1:IEND=NDUM
3395 CALL SITESCRN(ESN$,RV$):CALL SCRNDUMP(IST,IEND)
3400 NFUN=1:NFILE=4:CALL PRINTFILE(NFUN,NFILE)
3410 PRINT #4, " ":PRINT #4, " "
3420 FOR I=1 TO 80:PRINT #4, "=";:NEXT I:PRINT #4, "
    ":PRINT #4, " "
```

The next output generated contains the input data for the structure geometry screen. The output of the site specific data and the structure geometry screen are written to the first page of output. At the conclusion of this output, a form feed (CHR$(12)) character will be written to the data file. The source codes for this portion of the output are provided below:

```
3440 REM STRUCTURE GEOMETRY ON FIRST PAGE ALSO
3450 NPR=2:CALL GETLOC(NPR,DUM())
3460 FOR I=1 TO NAD(NPR):LPTS(I)=DUM(I):NEXT I
3470 CALL GETFLD(NPR,DUM())
3475 FOR I=1 TO NAD(NPR):FLD(I)=DUM(I):NEXT I
3480 FREC=IREC(NPR):CALL DSKRD(FREC,NDUM,DUM(),DUM$())
3490 FOR I=1 TO NDUM:VAR$(I)=DUM$(I):NEXT I
3500 IST=1:IEND=NDUM
3505 CALL GEOMSCRN(ESN$,RV$):CALL SCRNDUMP(IST,IEND)
3510 NFUN=1:NFILE=4:CALL PRINTFILE(NFUN,NFILE)
3520 PRINT #4, CHR$(12)
```

The next output generated is up to two screens of support member data. One of the two screens (NPR=3 or 4) will have at least one line of input and will be displayed. If either of the two screens contains no data, it will not be displayed. The source codes for this portion of the output are provided below:

```
3530 REM
3540 REM SUPPORT MEMBER DATA BEGINNING ON PAGE TWO
3550 NPR=3:NPG=0:NLOC=NAD(NPR)*15
3560 FREC=IREC(NPR):IF FREC=0 THEN 3740
3570 CALL GETLOC(NPR,DUM()):FOR I=2 TO 15:FOR J=1 TO NAD(NPR)
3580 IJ=(I-1)*NAD(NPR)+J:DUM(IJ)=DUM(J)+100*(I-1):NEXT J:NEXT I
3590 FOR I=1 TO NLOC:LPTS(I)=DUM(I):NEXT I
3600 CALL GETFLD(NPR,DUM()):FOR I=2 TO 15:FOR J=1 TO NAD(NPR)
3610 IJ=(I-1)*NAD(NPR)+J:DUM(IJ)=DUM(J):NEXT J:NEXT I
3620 FOR I=1 TO NLOC:FLD(I)=DUM(I):NEXT I
3630 CALL DSKRD(FREC,NDUM,DUM(),DUM$())
3640 FOR I=1 TO 15:IJ=(I-1)*NAD(NPR)+1
3650 A$=DUM$(IJ):CALL BLANK(A$,NBLK):IF NBLK>0 THEN 3670
3660 NEXT I:GOTO 3740
3670 FOR I=1 TO NDUM:VAR$(I)=DUM$(I):NEXT I
3680 CALL POLSCRN(ESN$,RV$)
3685 IST=1:IEND=NDUM:CALL SCRNDUMP(IST,IEND)
3690 NFUN=1:NFILE=4:CALL PRINTFILE(NFUN,NFILE)
3700 IF NPG=1 THEN PRINT #4, CHR$(12):NPG=0:GOTO 3740
3710 PRINT #4, " ":PRINT #4, " "
3720 FOR I=1 TO 80:PRINT #4, "=";:NEXT I:PRINT #4, "
     ":PRINT #4, " ":NPG=1
3730 REM
3740 IF NPR<4 THEN NPR=NPR+1:GOTO 3560
```

The next output generated is up to two screens of netting data. One of the two screens (NPR=5 or 6) will have at least one line of input and will be displayed. If either of the two screens contains no data, it will not be displayed. The source codes for this portion of the output are provided below:

```
3760 REM NETTING MATERIAL DATA
3770 NPR=5:NLOC=NAD(NPR)*15
3780 FREC=IREC(NPR):IF FREC=0 THEN 3960
3790 CALL GETLOC(NPR,DUM()):FOR I=2 TO 15:FOR J=1 TO NAD(NPR)
3800 IJ=(I-1)*NAD(NPR)+J:DUM(IJ)=DUM(J)+100*(I-1):NEXT J:NEXT I
3810 FOR I=1 TO NLOC:LPTS(I)=DUM(I):NEXT I
3820 CALL GETFLD(NPR,DUM()):FOR I=2 TO 15:FOR J=1 TO NAD(NPR)
3830 IJ=(I-1)*NAD(NPR)+J:DUM(IJ)=DUM(J):NEXT J:NEXT I
3840 FOR I=1 TO NLOC:FLD(I)=DUM(I):NEXT I
3850 CALL DSKRD(FREC,NDUM,DUM(),DUM$())
3860 FOR I=1 TO 15:IJ=(I-1)*NAD(NPR)+1
3870 A$=DUM$(IJ):CALL BLANK(A$,NBLK):IF NBLK>0 THEN 3890
3880 NEXT I:GOTO 3960
3890 FOR I=1 TO NDUM:VAR$(I)=DUM$(I):NEXT I
3900 CALL NETSCRN(ESN$,RV$)
3905 IST=1:IEND=NDUM:CALL SCRNDUMP(IST,IEND)
3910 NFUN=1:NFILE=4:CALL PRINTFILE(NFUN,NFILE)
3920 IF NPG=1 THEN PRINT #4, CHR$(12):NPG=0:GOTO 3960
3930 PRINT #4, " ":PRINT #4, " "
3940 FOR I=1 TO 80:PRINT #4, "=";:NEXT I:PRINT #4, " ":PRINT #4,
     " ":NPG=1
3950 REM
3960 IF NPR<6 THEN NPR=NPR+1:GOTO 3780
```

The next output generated is up to two screens of tension member data. One of the two screens (NPR=7 or 8) will have at least one line of input and will be displayed. If either of the two screens contains no data, it will not be displayed. The source codes for this portion of the output are provided below:

```
3980 REM TENSION MEMBER DATA
3990 NPR=7:NLOC=NAD(NPR)*15
4000 FREC=IREC(NPR):IF FREC=0 THEN 4180
4010 CALL GETLOC(NPR,DUM()):FOR I=2 TO 15:FOR J=1 TO NAD(NPR)
4020 IJ=(I-1)*NAD(NPR)+J:DUM(IJ)=DUM(J)+100*(I-1):NEXT J:NEXT I
4030 FOR I=1 TO NLOC:LPTS(I)=DUM(I):NEXT I
4040 CALL GETFLD(NPR,DUM()):FOR I=2 TO 15:FOR J=1 TO NAD(NPR)
4050 IJ=(I-1)*NAD(NPR)+J:DUM(IJ)=DUM(J):NEXT J:NEXT I
4060 FOR I=1 TO NLOC:FLD(I)=DUM(I):NEXT I
4070 CALL DSKRD(FREC,NDUM,DUM(),DUM$())
4080 FOR I=1 TO 15:IJ=(I-1)*NAD(NPR)+1
4090 A$=DUM$(IJ):CALL BLANK(A$,NBLK):IF NBLK>0 THEN 4110
4100 NEXT I:GOTO 4180
4110 FOR I=1 TO NDUM:VAR$(I)=DUM$(I):NEXT I
4120 CALL TENSCRN(ESN$,RV$)
4125 IST=1:IEND=NDUM:CALL SCRNDUMP(IST,IEND)
4130 NFUN=1:NFILE=4:CALL PRINTFILE(NFUN,NFILE)
```

```
4140 IF NPG=1 THEN PRINT #4, CHR$(12):NPG=0:GOTO 4180
4150 PRINT #4, " ":PRINT #4, " "
4160 FOR I=1 TO 80:PRINT #4, "=";:NEXT I:PRINT #4, "
     ":PRINT #4, " ":NPG=1
4170 REM
4180 IF NPR<8 THEN NPR=NPR+1:GOTO 4000
```

The next output generated is up to two screens of anchor data. One of the two screens (NPR=9 or 10) will have at least one line of input and will be displayed. If either of the two screens contains no data, it will not be displayed. When the last screen has been written to the data file, the routine is completed, and control is returned to the DROUTPUT routine. The source codes for this portion of the output are provided below:

```
4200 REM ANCHOR DATA
4210 NPR=9:NLOC=NAD(NPR)*15
4220 FREC=IREC(NPR):IF FREC=0 THEN 4400
4230 CALL GETLOC(NPR,DUM()):FOR I=2 TO 15:FOR J=1 TO NAD(NPR)
4240 IJ=(I-1)*NAD(NPR)+J:DUM(IJ)=DUM(J)+100*(I-1):NEXT J:NEXT I
4250 FOR I=1 TO NLOC:LPTS(I)=DUM(I):NEXT I
4260 CALL GETFLD(NPR,DUM()):FOR I=2 TO 15:FOR J=1 TO NAD(NPR)
4270 IJ=(I-1)*NAD(NPR)+J:DUM(IJ)=DUM(J):NEXT J:NEXT I
4280 FOR I=1 TO NLOC:FLD(I)=DUM(I):NEXT I
4290 CALL DSKRD(FREC,NDUM,DUM(),DUM$())
4300 FOR I=1 TO 15:IJ=(I-1)*NAD(NPR)+1
4310 A$=DUM$(IJ):CALL BLANK(A$,NBLK):IF NBLK>0 THEN 4330
4320 NEXT I:GOTO 4400
4330 FOR I=1 TO NDUM:VAR$(I)=DUM$(I):NEXT I
4340 CALL ANCSCRN(ESN$,RV$)
4345 IST=1:IEND=NDUM:CALL SCRNDUMP(IST,IEND)
4350 NFUN=1:NFILE=4:CALL PRINTFILE(NFUN,NFILE)
4360 IF NPG=1 THEN PRINT #4, CHR$(12):NPG=0:GOTO 4400
4370 PRINT #4, " ":PRINT #4, " "
4380 FOR I=1 TO 80:PRINT #4, "=";:NEXT I:PRINT #4, "
     ":PRINT #4, " ":NPG=1
4390 REM
4400 IF NPR<10 THEN NPR=NPR+1:GOTO 4220
4410 IF NPG=1 THEN PRINT #4, CHR$(12)
4420 REM
4430 CLOSE
4440 END SUB
```

PRINTFILE Routine. The PRINTFILE routine allows the current display on the monitor to be: (1) appended to a designated file device, (2) dumped to the printer, or (3) both appended to a file and dumped to the printer. The function number (1-3) designates what action is desired, and the file number designates the device number of the desired output file. On each CALL to the PRINTFILE routine, the function number (NFUN) is specified as 1 (append screen display to a designated data file), and the file device number (NFIL) is specified as 4 (.PRT sequential output file name). The source codes for the PRINTFILE routine are provided below:

```
2770 SUB PRINTFILE(NFUN, NFIL)
2780 REM THIS SUBROUTINE IS TO SAVE THE SCREEN TO A FILE AND/OR
2790 REM DUMP THE SCREEN TO THE PRINTER
2800 REM
2810 REM NFUN = THE FUNCTION OPTION NUMBER
2820 REM FILE$= THE FILE NAME TO SAVE SCREEN DATA
2830 REM
2840 REM NFIL = THE STARTING BUFFER NUMBER TO OPEN FILES
2850 REM NROW = THE TOTAL NUMBER OF ROWS ON THE SCREEN
2860 REM NCOL = THE TOTAL NUMBER OF COLUMNS ON THE SCREEN'
2870 REM PAGE$= THE FLAG TO FEED NEW FORM (Y/N)
2880 REM
2890 ON ERROR GOTO CLOSEFILE
2900 IF NFUN < 1 OR NFUN > 3 THEN
2910 CLS: PRINT "ERROR FOUND IN FUNCTION OPTION NUMBER"
2920 PRINT "1. APPEND THE SCREEN TO A FILE"
2930 PRINT "2. DUMP   THE SCREEN TO PRINTER"
2940 PRINT "3. BOTH OF FUNCTION 1. AND FUNCTION 2."
2950 INPUT "PRESS ANY KEY TO CONTINUE . . . ",ANS$
2960 GOTO STOPEXECUTION
2970 END IF
2980 NROW=24:NCOL=80
2990 M = NFIL
3000 FOR I = 1 TO NROW
3010 FOR J = 1 TO NCOL
3020 IF NFUN=1 OR NFUN=3 THEN
3030   IF J=NCOL THEN
3040     PRINT #M,CHR$(SCREEN(I,J))
3050   ELSE
3060     PRINT #M,CHR$(SCREEN(I,J));
3070   END IF
3080 END IF
3090 IF NFUN=2 OR NFUN=3 THEN
3100   PRINT #NFIL,CHR$(SCREEN(I,J));
3110 END IF
3120 NEXT J
3130 NEXT I
3140 REM
3150 STOPEXECUTION:
3160 END SUB
3170 REM
3180 CLOSEFILE:
3190 RESUME NEXT
```

Structural Design. The results of the structural design (see Chapter 6 DESIGN Routine) consists of the following: (1) the deflections for the top of each support member, and (2) a list of the tension members used to provide lateral support between the support members, and guyed support of the exterior support members.

The node numbers of the structural system are numbered at the support members as follows: the lower left support member is node number 1; from the lower left support member, succeeding exterior support members to the right (along the width of the structure) are numbered sequentially until the lower right support member is numbered. The next numbered support member is the left-most member on the second bay (along the length of the structure) of supports. Numbering continues from that support to the right along the width of the structure to the right-most member. Numbering continues up the bays and across the width of the structure until the upper right support member is numbered. An example of node numbering for a structure with two 20-foot bays and three 10-foot member spacings is shown in Figure 7 below:

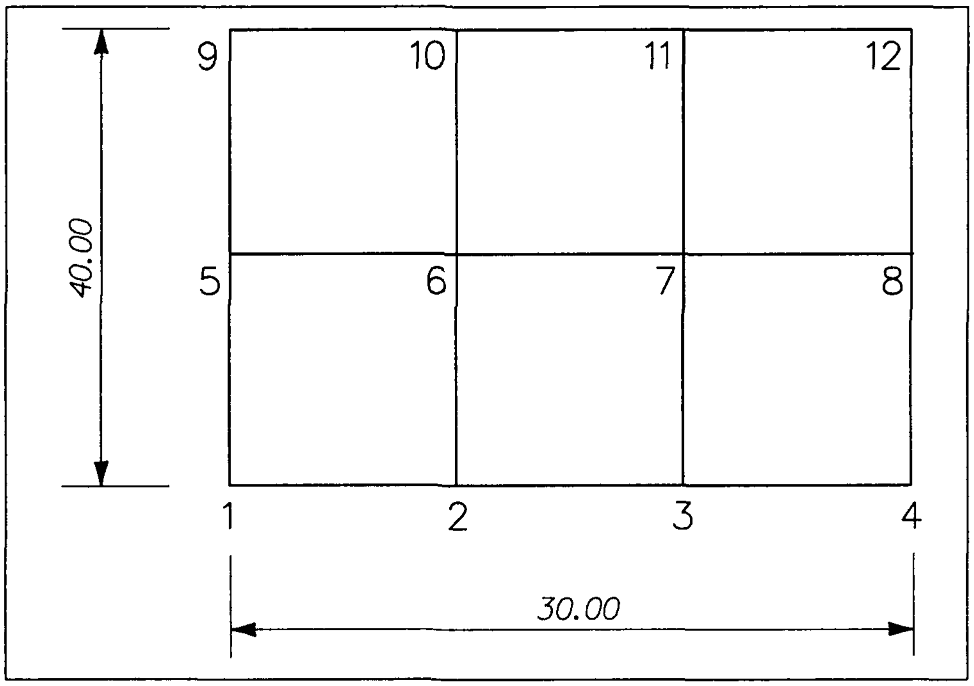

Figure 7. Node number scheme for tension structures

Before the remainder of the output forms are provided, two data items must be extracted from the project data file. These two data items are the specified height of the structure and the specified guy condition for the structure. Both of these data items are contained on the structure geometry screen (NPR=2). The structure height (HTPOL) is data item number 24, and the guy condition (GUY$) is data item number 25. The source codes required to defined the project data file name, OPEN the project data file, OPEN the screen field length data file (DATA.FLD), and extract these two data items are provided below:

```
210 REM
220 REM LOAD HEIGHT OF STRUCTURE AND GUY CONDITION
230 FL$="P"+MID$(ESN$,1,6)+MID$(RV$,1,1)+".DAT"
240 OPEN DD$+FL$ AS #1 LEN=128:FIELD #1, 128 AS AA$
250 OPEN DD$+"DATA.FLD" AS #3 LEN=4:FIELD #3, 4 AS DF$
260 REM STRUCTURE GEOMETRY
270 NPR=2:CALL GETFLD(NPR,DUM())
280 FREC=IREC(NPR):CALL DSKRD(FREC,NDUM,DUM(),DUM$())
290 A$=DUM$(24):CALL SCAN(A$,V(),NN,B$(),NW):HTPOL=V(1)*12
300 GUY$=MID$(DUM$(25),1,1)
310 CLOSE #3:CLOSE #1
```

The geometry of the structure must be defined next. The bay spacings and derived pole spacings along the width dimension of the structure are needed to determine the number of support members and tension members included in the structure. The bay spacings were written to the DESINPUT.DAT sequential data file to serve as the input to the DESIGN routine. The first record of the data file contains the number of bay spacings (NBAY), and the individual bay spacings (BAYS()) are contained on succeeding records (one bay spacing per record). The pole spacings along the width dimension of the project were determined by the DESIGN routine and written to the DESOUTPT.DAT sequential data file. The number of pole spacings (NPOL) is contained in the first record, and the value of each pole spacing (PSPAC) is contained on the second record. The source codes required to extract these data items are provided below:

```
350 OPEN "DESINPUT.DAT" FOR INPUT AS #1
360 LINE INPUT #1, A$:CALL SCAN(A$,V(),NN,B$(),NW):IF V(1)=0
    THEN 1800
370 NBAY=V(1):FOR IZ=1 TO NBAY
380 LINE INPUT #1, A$:CALL SCAN(A$,V(),NN,B$(),NW):BAYS(IZ)=V(1)
390 NEXT IZ:CLOSE #1
400 REM
410 OPEN "DESOUTPT.DAT" FOR INPUT AS #1
420 LINE INPUT #1, A$:CALL SCAN(A$,V(),NN,B$(),NW):IF V(1)=0
    THEN 1800
430 NPOL=V(1):LINE INPUT #1, A$:CALL SCAN(A$,V(),NN,B$(),NW)
440 IF V(1)=0 THEN 1800
450 PSPAC=V(1):CLOSE #1
```

Support Member Deflections. The top-of-member deflections are calculated in the DESIGN routine and stored in the sequential data file DEFLECTS.DAT. The DESIGN routine writes two sequential records to the data file for each support member: the first record contains the deflection in the length dimension (along the bays), and the second record contains the deflection in the width dimension (along the pole spacings). The support members are processed sequentially in the DESIGN routine, and the deflections are written to the DEFLECTS.DAT file as each member is being analyzed. The data file is OPENed, two records are read for each support member, and the data are written to the output file. When the output is complete, the DEFLECTS.DAT data file is CLOSEd and erased (KILLed). An example of the support member deflections output is provided in Appendix C. The source codes for this output are provided below:

```
470 REM GEOMETRIC DATA EXTRACTED SUCCESSFULLY - BEGIN OUTPUT
480 REM   PRINTER FORM ASSUMED TO BE TOP-OF-PAGE
490 OPEN PL$ FOR APPEND AS #4
500 REM
510 REM TOP OF SUPPORT MEMBER DEFLECTIONS
520 PRINT #4, SPC(50);:PRINT #4, USING " PROJECT:  \    \";ESN$
530 PRINT #4, SPC(50);:PRINT #4, USING " REVISION:  \\";RV$
535 PRINT #4, SPC(50);:PRINT #4, USING " PROG VER:  \  \";PGVER$
540 PRINT #4, SPC(50);:PRINT #4, USING " PROG NUM:  ###";DLNUM
545 PRINT #4, " "
550 PRINT #4, SPC(16);
555 PRINT #4, "STRUCTURAL SYSTEM SUPPORT MEMBER DEFLECTIONS"
560 PRINT #4, " "
570 A$="SUPPORT         TOP OF MEMBER DEFLECTION IN THE"
580 PRINT #4, SPC(10);:PRINT #4, A$
590 A$="MEMBER      LENGTH DIMENSION      WIDTH DIMENSION"
600 PRINT #4, SPC(11);:PRINT #4, A$
610 A$="NUMBER           (INCHES)             (INCHES)"
620 PRINT #4, SPC(11);:PRINT #4, A$
630 A$="         ###          ###.######          ###.######"
640 OPEN "DEFLECTS.DAT" FOR INPUT AS #1
650 NLINE=0:CABTOT=0!:GUYTOT=0!
660 IF EOF(1) THEN 700
670 NLINE=NLINE+1
675 LINE INPUT #1,X$:CALL SCAN(X$,V(),NN,B$(),NW):XDF=V(1)
680 LINE INPUT #1,X$:CALL SCAN(X$,V(),NN,B$(),NW):YDF=V(1)
690 PRINT #4, USING A$;NLINE,YDF,XDF:GOTO 660
700 PRINT #4, CHR$(12):CLOSE #1:KILL "DEFLECTS.DAT"
```

Tension Member Data. The tension members are categorized as either (a) exterior width, (b) exterior length, (c) interior width, (d) interior length, (e) diagonal, (f) horizontal guy, or (g) vertical guy members. For each length of tension member used, the following data are provided:

1. FROM node number, origin point of the member,

2. TO node number, termination point of the member,

3. Actual (measured) length of the member section, inches,

4. Unstressed (pull-to) length of the member section, inches, and

5. Category type of the member section.

The actual or measured length of each tension member is calculated from the bay spacings (along the structure length) and pole spacings (along the structure width). The number and value of the bay spacings are stored in the DESIGN routine sequential input data file DESINPUT.DAT. The number and value of the pole spacings are stored in the DESIGN routine sequential output data file named DESOUTPT.DAT. The un-stressed or pull-to lengths are calculated in the DESIGN routine and are also written to the DESOUTPT.DAT data file. In addition to the un-stressed length data, structure node numbers and tension member types are also written to the DESOUTPT.DAT data file. An example of the file contents is provided below:

```
  3
      80.0000
   22175.5977
     219.1895
     156.6380          1          5        1.00
     175.8413          1          6        5.00
      92.8664          1          2        3.00
     175.8413          2          5        5.00
     156.6398          2          6        2.00
     175.8413          2          7        5.00
      92.8664          2          3        3.00
     175.8413          3          6        5.00
     156.6398          3          7        2.00
     175.8413          3          8        5.00
      92.8664          3          4        3.00
     175.8413          4          7        5.00
     156.6380          4          8        1.00
     156.6380          5          9        1.00
     175.8413          5         10        5.00
      92.8668          5          6        4.00
     175.8413          6          9        5.00
     156.6398          6         10        2.00
     175.8413          6         11        5.00
      92.8668          6          7        4.00
     175.8413          7         10        5.00
     156.6398          7         11        2.00
     175.8413          7         12        5.00
      92.8668          7          8        4.00
     175.8413          8         11        5.00
     156.6380          8         12        1.00
      92.8664          9         10        3.00
      92.8664         10         11        3.00
      92.8664         11         12        3.00
     169.7070          1      -1000        6.00
```

The first record (line) contains the number of pole spacings required for the satisfactory structural design. The second record contains the pole spacing in inches. From these data, three pole spacings were required at

80.0 inches per spacing, thus yielding a total structure width of 240.0 inches or 20.0 feet. The third record is the maximum moment developed at the base of a support member in inch-pounds. The fourth record is the maximum tension developed in the tension members in pounds. The remaining records contain the unstressed length, from and to node numbers and classification of each tension member in the design. The last record contains the unstressed length of the tension members used as guys. The actual length of each guy is measured along a 45-degree angle from the top of the support member to the ground surface. The tension member classifications are as follows (see Chapter 6 DESIGN Routine):

1. Bay length exterior
2. Bay length interior
3. Pole spacing exterior
4. Pole spacing interior
5. Diagonal
6. Bottom row (vertical) guys
7. Top row (vertical) guys
8. Left-most column (horizontal) guys
9. Right-most column (horizontal) guys

At the conclusion of this output, the sequential data files, DESINPUT.DAT and DESOUTPT.DAT, are erased (KILLed) from the hard disk. An example of the structural system tension member output is provided in Appendix D. The source codes for this output are provided below:

```
720 REM  STRUCTURE TENSION MEMBER LENGTH AND TYPE OUTPUT
730 GOSUB 760:GOTO 880
740 REM
750 REM HEADINGS FOR TENSION MEMBER OUTPUT
760 PRINT #4, SPC(50);:PRINT #4, USING "  PROJECT:  \     \";ESN$
770 PRINT #4, SPC(50);:PRINT #4, USING " REVISION:  \\";RV$
775 PRINT #4, SPC(50);:PRINT #4, USING " PROG VER:  \  \";PGVER$
780 PRINT #4, SPC(50);:PRINT #4, USING " PROG NUM:  ###";DLNUM
785 PRINT #4, " "
790 PRINT #4, SPC(19);
795 PRINT #4, "STRUCTURAL SYSTEM TENSION MEMBER LENGTHS"
800 PRINT #4, " ":NLINE=0
810 PRINT #4, SPC(17);:PRINT #4, " MIN DISTANCE  UNSTRESSED
    LENGTH"
```

```
820 A$="FROM    TO    (INCHES)          (INCHES)          CABLE
    TYPE"
830 PRINT #4, SPC(5);:PRINT #4, A$
840 A$="       ###    ###    #####.##            #####.##
              \        \ \     \"
850 RETURN
860 REM
870 REM HORIZONTAL CABLES FIRST
880 FOR IZ=1 TO NBAY+1:NST=(IZ-1)*(NPOL+1)
890 STYP$="WIDTH"
900 CCLAS=4:IF IZ=1 OR IZ=NBAY+1 THEN CCLAS=3
910 FOR JZ=1 TO NPOL:NFM=NST+JZ:NTO=NFM+1:ADIST=PSPAC
920 OPEN "DESOUTPT.DAT" FOR INPUT AS #1
930 FOR KZ=1 TO 4:LINE INPUT #1, X$:NEXT KZ
940 IF EOF(1) THEN 990
950 LINE INPUT #1, X$:CALL SCAN(X$,V(),NN,B$(),NW)
960 IF CCLAS<>V(4) THEN 940
970 IF NFM<>V(2) OR NTO<>V(3) THEN 940
980 UDIST=V(1):CABTOT=CABTOT+UDIST
990 IF CCLAS=3 THEN FTYP$="EXTERIOR"
1000 IF CCLAS=4 THEN FTYP$="INTERIOR"
1010 CLOSE #1
1015 IF NLINE>50 THEN PRINT #4, CHR$(12):GOSUB 760
1020 PRINT #4, USING A$;NFM,NTO,ADIST,UDIST,FTYP$,STYP$
1030 NLINE=NLINE+1:NEXT JZ:NEXT IZ
1040 REM
1050 REM VERTICAL CABLES NEXT
1060 FOR IZ=1 TO NPOL+1:FOR JZ=1 TO NBAY
1070 CCLAS=2:IF IZ=1 OR IZ=NPOL+1 THEN CCLAS=1
1080 STYP$="LENGTH"
1090 NFM=(JZ-1)*(NPOL+1)+IZ:NTO=NFM+NPOL+1:ADIST=BAYS(JZ)*12
1100 OPEN "DESOUTPT.DAT" FOR INPUT AS #1
1110 FOR KZ=1 TO 4:LINE INPUT #1, X$:NEXT KZ
1120 IF EOF(1) THEN 1170
1130 LINE INPUT #1, X$:CALL SCAN(X$,V(),NN,B$(),NW)
1140 IF CCLAS<>V(4) THEN 1120
1150 IF NFM<>V(2) OR NTO<>V(3) THEN 1120
1160 UDIST=V(1):CABTOT=CABTOT+UDIST
1170 IF CCLAS=1 THEN FTYP$="EXTERIOR"
1180 IF CCLAS=2 THEN FTYP$="INTERIOR"
1190 CLOSE #1
1195 IF NLINE>50 THEN PRINT #4, CHR$(12):GOSUB 760
1200 PRINT #4, USING A$;NFM,NTO,ADIST,UDIST,FTYP$,STYP$
1210 NLINE=NLINE+1:NEXT JZ:NEXT IZ
1220 REM
1230 REM DIAGONAL CABLES
1240 FTYP$="DIAGONAL":STYP$=" ":CCLAS=5
1250 FOR IZ=1 TO NPOL:FOR JZ=1 TO NBAY
1260 NST=(JZ-1)*(NPOL+1)+IZ:NFM=NST:NTO=NFM+NPOL+2
1270 ADIST=SQR((BAYS(JZ)*12)^2+FSPAC^2)
```

```
1280 OPEN "DESOUTPT.DAT" FOR INPUT AS #1
1290 FOR KZ=1 TO 4:LINE INPUT #1, X$:NEXT KZ
1300 IF EOF(1) THEN 1350
1310 LINE INPUT #1, X$:CALL SCAN(X$,V(),NN,B$(),NW)
1320 IF CCLAS<>V(4) THEN 1300
1330 IF NFM<>V(2) OR NTO<>V(3) THEN 1300
1340 UDIST=V(1):CABTOT=CABTOT+UDIST
1350 CLOSE #1
1360 IF NLINE>50 THEN PRINT #4, CHR$(12):GOSUB 760
1370 PRINT #4, USING A$;NFM,NTO,ADIST,UDIST,FTYP$,STYP$
1380 NLINE=NLINE+1:NTO=NFM+NPOL+1:NFM=NFM+1
1390 OPEN "DESOUTPT.DAT" FOR INPUT AS #1
1400 FOR KZ=1 TO 4:LINE INPUT #1, X$:NEXT KZ
1410 IF EOF(1) THEN 1460
1420 LINE INPUT #1, X$:CALL SCAN(X$,V(),NN,B$(),NW)
1430 IF CCLAS<>V(4) THEN 1410
1440 IF NFM<>V(2) OR NTO<>V(3) THEN 1410
1450 UDIST=V(1):CABTOT=CABTOT+UDIST
1460 CLOSE #1
1470 IF NLINE>50 THEN PRINT #4, CHR$(12):GOSUB 760
1480 PRINT #4, USING A$;NFM,NTO,ADIST,UDIST,FTYP$,STYP$
1490 NLINE=NLINE+1:NEXT JZ:NEXT IZ
1500 REM
1510 REM GUYS
1520 IF GUY$<>"Y" THEN 1810
1530 FTYP$="HORZ":STYP$="GUY"
1540 OPEN "DESOUTPT.DAT" FOR INPUT AS #1
1550 FOR KZ=1 TO 4:LINE INPUT #1, X$:NEXT KZ
1560 IF EOF(1) THEN 1800
1570 LINE INPUT #1, X$
1575 CALL SCAN(X$,V(),NN,B$(),NW):IF V(4)<>6! THEN 1560
1580 UDIST=V(1):ADIST=SQR(2*HTPOL^2)
1590 FOR IZ=1 TO NBAY+1:NFM=(IZ-1)*(NPOL+1)+1:NTO=NFM
1600 IF NLINE>50 THEN PRINT #4, CHR$(12):GOSUB 760
1610 PRINT #4, USING A$;NFM,NTO,ADIST,UDIST,FTYP$,STYP$
1620 GUYTOT=GUYTOT+UDIST
1630 NLINE=NLINE+1:NEXT IZ
1640 FTYP$="VERT"
1650 FOR IZ=1 TO NPOL+1:NFM=IZ:NTO=NFM
1660 IF NLINE>50 THEN PRINT #4, CHR$(12):GOSUB 760
1670 PRINT #4, USING A$;NFM,NTO,ADIST,UDIST,FTYP$,STYP$
1680 GUYTOT=GUYTOT+UDIST
1690 NLINE=NLINE+1:NEXT IZ
1700 FTYP$="HORZ":FOR IZ=1 TO NBAY+1:NFM=(IZ-1)*(NPOL+1)+NPOL+1
1710 NTO=NFM:IF NLINE>50 THEN PRINT #4, CHR$(12):GOSUB 760
1720 PRINT #4, USING A$;NFM,NTO,ADIST,UDIST,FTYP$,STYP$
1730 GUYTOT=GUYTOT+UDIST
1740 NLINE=NLINE+1:NEXT IZ
1750 FTYP$="VERT":NST=NBAY*(NPOL+1)
1755 FOR IZ=1 TO NPOL+1:NFM=NST+IZ
```

```
1760 NTO=NFM:IF NLINE>50 THEN PRINT #4, CHR$(12):GOSUB 760
1770 PRINT #4, USING A$;NFM,NTO,ADIST,UDIST,FTYP$,STYP$
1780 GUYTOT=GUYTOT+UDIST
1790 NLINE=NLINE+1:NEXT IZ
1800 CLOSE #1
1810 KILL "DESOUTPT.DAT":KILL "DESINPUT.DAT"
1820 PRINT #4, CHR$(12)
```

Foundation/Footing Design. The results of the foundation/footing design are written to the FOOTING.DAT sequential data file (see Chapter 7 FOOTING Routine). The FOOTING.DAT data file is simply OPENed, and its contents are printed to the output .PRT data file. When the data transfer is completed, the FOOTING.DAT data file is erased (KILLed). An example of the foundation/footing design output is provided in Appendix D. The source codes for the foundation/footing design output are provided below:

```
1840 REM FOUNDATION/FOOTING OUTPUT (FOOTING.DAT)·
1850 PRINT #4, SPC(50);:PRINT #4, USING " PROJECT:  \    \";ESN$
1860 PRINT #4, SPC(50);:PRINT #4, USING " REVISION:  \\";RV$
1865 PRINT #4, SPC(50);:PRINT #4, USING " PROG VER:  \  \";PGVER$
1870 PRINT #4, SPC(50);:PRINT #4, USING " PROG NUM:  ###";DLNUM
1875 PRINT #4, " ":PRINT #4, " "
1880 PRINT #4, SPC(23);:PRINT #4, "FOUNDATION/FOOTING SYSTEM
     DESIGN"
1890 PRINT #4, " "
1900 OPEN "FOOTING.DAT" FOR INPUT AS #1
1910 IF EOF(1) THEN 2010
1920 LINE INPUT #1, A$:PRINT #4, A$
1930 IF LEN(A$)<=6 THEN 1910
1940 FOR IZ=1 TO LEN(A$)-6
1950 IF MID$(A$,IZ,6)="LENGTH" THEN
1960      CALL SCAN(A$,V(),NN,B$(),NW):DEPTH=V(1)
1970      GOTO 1910
1980 ENDIF
1990 NEXT IZ
2000 GOTO 1910
2010 CLOSE #1:KILL "FOOTING.DAT"
2020 PRINT #4, " ":PRINT #4, " ":PRINT #4, " "
```

Anchor Design. The results of the anchor design are written to the ANCHOR.DAT sequential data file (see Chapter 8 ANCDES Routine). The ANCHOR.DAT data file is OPENed and its contents are printed to the output .PRT data file. When the data transfer is completed, the ANCHOR.DAT data file is erased (KILLed). An example of the anchor design output is provided in Appendix D. The source codes for the anchor design output are provided below:

```
2040 REM ANCHOR DESIGNS
2050 IF GUY$="Y" OR GUY$="y" THEN
```

```
2060        PRINT #4, SPC(23);
2065        PRINT #4, "SELECTED ANCHORING SYSTEM DESIGN"
2070        PRINT #4, " "
2080        OPEN "ANCHOR.DAT" FOR INPUT AS #1
2090        IF EOF(1) THEN 2110
2100        LINE INPUT #1, A$:PRINT #4, A$:GOTO 2090
2110        CLOSE #1:KILL "ANCHOR.DAT"
2120        PRINT #4, " ":PRINT #4, " ":PRINT #4, " "
2130 ENDIF
```

Materials Summary. The final output form consists of a summary of each structural material required by the designs. The support member data, netting data, tension member data for support members and guy members, and anchor data are provided in a short summary table. For each data item listed, the quantity, name, use code, description, unit cost, and total cost are provided. An example of the structural material summary is provided in Appendix D. The source codes for the material summary output are provided below:

```
2150 REM FINAL STRUCTURAL MATERIALS SUMMARY
2160 PRINT #4, SPC(24);:PRINT #4, "STRUCTURAL MATERIALS SUMMARY"
2170 PRINT #4, " "
2180 A$="QUANTITY    MARK   USE    DESCRIPTION        "
2185 A$=A$+"UNIT COST    TOTAL COST"
2190 PRINT #4, SPC(6);:PRINT #4, A$
2200 B$="  ####.   \   \    \\    \              \    #####.##
        #####.##"
2210 A$="      "+B$
2220 REM
2230 REM SUPPORT MEMBERS
2240 TOTPOL=HTPOL+DEPTH:NPOLE=(NBAY+1)*(NPOL+1)
2250 NCUT=1:IF TOTPOL=POLEN THEN NCUT=0
2260 IF TOTPOL<POLEN/2 THEN
2270        NCUT=2:NPOLE=NPOLE/2
2280        IF TOTPOL*2=POLEN THEN NCUT=1
2290 ENDIF
2300 MRK$=MID$(POLMK$,1,5):US$=MID$(POLMK$,6,1)
2310 CTOT=NPOLE*POLEN*PCOST
2320 PRINT #4, USING A$;NPOLE,MRK$,US$,POLDS$,PCOST,CTOT
2330 REM
2340 REM NETTING
2350 REM ASSUME OVERLAP OF HTPOL/2 ON ALL FOUR SIDES
2360 REM ASSUME FIVE (5.0) PERCENT OVERLAP BETWEEN NETS
2370 TOTL=0!:FOR IZ=1 TO NBAY:TOTL=TOTL+BAYS(IZ):NEXT IZ
2380 TOTW=NPOL*PSPAC/12!
2390 TAREA=(TOTL+HTPOL)*(TOTW+HTPOL)
2400 NAREA=NSIZE*.8:NNET=INT(TAREA/NAREA)+1!
2410 MRK$=MID$(NETMK$,1,5):US$=MID$(NETMK$,6,1)
2420 CTOT=NNET*NSIZE*NCOST
2430 PRINT #4, USING A$;NNET,MRK$,US$,NETDS$,NCOST,CTOT
```

```
2440 REM
2450 REM TENSION MEMBERS (STRUCTURE CABLES FIRST - GUYS SECOND)
2460 CABTOT=INT(CABTOT)+1!
2465 IF GUYTOT>0 THEN GUYTOT=INT(GUYTOT)+1!
2470 MRK$=MID$(CABMK$,1,5):US$=MID$(CABMK$,6,1)
2480 CTOT=CABTOT*CCOST
2490 PRINT #4, USING A$;CABTOT,MRK$,US$,CABDS$,CCOST,CTOT
2500 IF GUY$="Y" OR GUY$="y" THEN
2510      CTOT=GUYTOT*CCOST
2520      PRINT #4, USING A$;GUYTOT,MRK$,US$,CABDS$,CCOST,CTOT
2530 ENDIF
2540 REM
2550 REM ANCHORS FOR GUYED SYSTEM
2560 IF GUY$="Y" OR GUY$="y" THEN
2570      NANCH=(NBAY+1)*2+(NPOL+1)*2
2580      FOR IZ=1 TO NANC
2585      MRK$=MID$(ANCMK$(IZ),1,5):US$=MID$(ANCMK$(IZ),6,1)
2590      CTOT=ACOST(IZ)*NANCH
2600      PRINT #4, USING A$;NANCH,MRK$,US$,ANCDS$(IZ),
          ACOST(IZ),CTOT
2610      NEXT IZ
2620 ENDIF
2630 REM
2640 REM NOTES OF ASSUMPTIONS FOR MATERIAL SUMMARY
2650 PRINT #4, " ":PRINT #4, " "
2660 A$="NOTE:   ## CUT(S) REQUIRED ON EACH SUPPORT MEMBER"
2670 PRINT #4, SPC(10);:PRINT #4, USING A$;NCUT
2680 A$="NETTING OVERHANG IS ONE-HALF STRUCTURE HEIGHT"
2685 A$=A$+" ON ALL SIDES"
2690 PRINT #4, SPC(22);:PRINT #4, A$
2700 A$="            NETTING OVERLAP IS 5.0 PERCENT ON ALL SIDES"
2710 PRINT #4, SPC(10);:PRINT #4, A$
2720 PRINT #4, CHR$(12)
2730 CLOSE
2740 END SUB
```

Final Output Display. When all of the output forms have been generated, the sequential output .PRT data file will be displayed by the **UTILITY** routine. The UTILITY routine allows the data file to be: (1) browsed (reviewed) on the screen, (2) printed to the printer, (3) copied to a floppy disk (assumed to reside in the A disk drive), or (4) erased from the resident (hard) disk drive when the UTILITY routine is terminated. If the .PRT data file is not erased at the termination of the UTILITY routine, it will remain on the resident (hard) disk drive. This is the last phase of the program output. When this phase is concluded, the project data file and the DESIGN.DAT data files residing on the hard disk are erased (KILLed), and program control is returned to the original program prompt. The source codes for these final program operations are provided below:

```
3760 REM PROGRAM OUTPUT TO DISK FILE
3770 CALL DROUTPUT(ESN$,RV$,DD$,DLNUM,PGVER$)
3780 PL$=LEFT$(FL$,LEN(FL$)-3)+"PRT"
3790 A$="UTILITY "+PL$:SHELL A$
3800 KILL "DESIGN.DAT":KILL FL$
3810 GOTO 370
```

10 Program Compilation and Linking

The Camouflage Program contains a series of stand-alone executable program modules which allow the description and analysis of a camouflage structure. All of the program modules except two are written in BASIC, the two exceptions being the structural design routine (DESIGN) and the plan view graphics routine (DRAW) which are written in FORTRAN. The purpose of this chapter is to describe each executable program as to the compilation and linking procedures required to create that program module. The executable programs described are as follows: (1) MAIN (Camouflage), (2) VALID, (3) DEVICE, (4) UTILITY, (5) CAMOBLDR, (6) EDITPRIC, (7) DESIGN, and (8) DRAW.

MAIN Program. The Camouflage Program executable program module is named MAIN.EXE. The MAIN.EXE executable program consists of four program modules: MAIN, CAMINP, CAMOSCRN, and DROUTPUT. In addition, a library of general usage routines are stored in the program module INPUTROU. Each of these five modules is written in BASIC (Microsoft 1989A and B). The five routines are compiled using the Microsoft PRO-Basic Compiler (version 7.1). The library module GENERAL.LIB is created using the Quick Library option of the PRO-Basic Linker (version 5.10) and the PRO-Basic Library Manager (version 3.17). The final executable program MAIN.EXE is created by linking the object (OBJ) codes for the MAIN routine with the object codes for the three subprogram modules and the GENERAL.LIB library codes. The program linking is accomplished with the PRO-Basic Linker. The PRO-Basic compiler/linker/library commands required to produce the GENERAL.LIB library module are provided below:

```
ERASE GENERAL.*
BC INPUTROU.BAS /Fs;
LINK /Q INPUTROU,GENERAL.QLB,NUL,QBXQLB.LIB;
LIB GENERAL.LIB+INPUTROU;
ERASE INPUTROU.OBJ
ERASE *.QLB
```

The Far strings (Fs) compiler option is used to conserve string space in the main program module. This option provides string storage outside the DGROUP allowing more available storage for program variables, etc., in the DGROUP. The ERASE commands are MS-DOS (Microsoft 1989A and B) system commands which are used to erase the object (OBJ) modules when the linking process is completed. These commands are used to conserve the available space on the hard disk of the computer system. The PRO-Basic compiler/linker commands required to produce the MAIN.EXE executable module are provided below:

```
BC MAIN.BAS /X/D/O/Fs;
BC CAMINP.BAS /D/O/Fs;
BC CAMOSCRN.BAS /D/O/Fs;
BC DROUTPUT.BAS /O/D/X/Fs;
LINK /EX /NOE MAIN+(CAMINP+CAMOSCRN)+(DROUTPUT),,,GENERAL;
ERASE MAIN.OBJ
ERASE CAMINP.OBJ
ERASE CAMOSCRN.OBJ
ERASE DROUTPUT.OBJ
ERASE *.MAP
```

The compiler options used are for error checking and resume recoveries (X), debug (D, allows for <CTRL BREAK> interrupts), stand-alone libraries (O), and far strings (Fs). The linker options used are for packing executables (EX) and ignoring extending dictionaries (NOE). The use of these two options results in smaller executable modules. During the linker process, the input/edit program function modules, CAMINP and CAMOSCRN, are overlaid with the output program function module, DROUTPUT. This overlaid program executable results in a smaller memory (RAM) requirement for program operation. The ERASE commands are used to erase the object modules and the mapping module from the hard disk when the linking process is completed.

VALID Program. The VALID program is a single program module which is linked to the program libraries module GENERAL. The compiler options used are for error checking only (E), far strings, floating point operations (FPi), stand-alone libraries, and debug. The PRO-Basic compiler/linker commands required to produce the VALID.EXE executable module are provided below:

```
BC VALID.BAS /E/FS/FPi/O/D;
LINK /EX /NOE VALID,,,GENERAL;
ERASE VALID.OBJ
```

DEVICE Program. The DEVICE program is a single program module which is linked to the program libraries module GENERAL. The compiler options used are for error checking only (E), far strings, floating point operations (FPi), stand-alone libraries, and debug. The PRO-Basic compiler/linker commands required to produce the DEVICE.EXE executable module are provided below:

```
BC DEVICE.BAS /E/FS/FPi/O/D;
LINK /EX /NOE DEVICE,,,GENERAL;
ERASE DEVICE.OBJ
```

UTILITY Program. The UTILITY program is a single program module written by Mr. Chong Lin for Gulf States Manufacturers, Inc., of Starkville, MS. The program module was written in BASIC and compiled using the Microsoft Quick-BASIC compiler (version 4.5). Source codes are unavailable for this program module.

CAMOBLDR Program. The CAMOBLDR program is a stand-alone program module that is totally self-contained. The compiler options used are stand-alone library (O) and far string storage (Fs). The Pro-Basic compiler/linker commands required to produce the CAMOBLDR.EXE executable module are provided below:

```
BC CAMOBLDR.BAS /O/Fs;
LINK /EX /NOE CAMOBLDR;
ERASE CAMOBLDR.OBJ
```

EDITPRIC Program. The EDITPRIC program is a single program module which is linked to the program libraries module GENERAL. The compiler options used are far strings, stand-alone libraries, and debug. The PRO-Basic compiler/linker commands required to produce the EDITPRIC.EXE executable module are provided below:

```
BC EDITPRIC.BAS /Fs/O/D;
LINK /EX /NOE EDITPRIC,EDITPRIC,,GENERAL;
ERASE EDITPRIC.OBJ
ERASE *.MAP
```

DESIGN Program. The DESIGN program is a single program module written in FORTRAN (Ryan-McFarland 1987). The program executable is created using the RM FORTRAN compiler and linker (version 2.4). The main routine and 24 subroutines are in a single module. The compilation and linking are performed within the RMFORTRAN environment. The RMFORTRAN linker commands required to produce the DESIGN.EXE executable module are provided below:

```
OUTPUT DESIGN.EXE
MAP=   DESIGN.MAP
FILE   DESIGN.OBJ
```

DRAW Program. The DRAW program consists of a main routine (DRAW) and four subroutines (BORDER, CIRCLE, DIMLIN, and FOOTS). Each of the routines is written in FORTRAN. The program executable is created using the RM FORTRAN compiler and linker (version 2.4). The graphics used are PLOT88 utilities (Plotworks 1989) which are linked with the plot driver utilities as library routines. In addition, the PLOT88 routines, HPNULL, HINULL, and PSNULL, are linked to move the drivers for plotter devices. The compilations and linking are performed within the RMFORTRAN environment. The RMFORTRAN linker commands required to produce the DRAW.EXE executable module are provided below:

```
OUTPUT DRAW.EXE
LIBRARY PLOT88,DRIVE88
MAP=   DRAW.MAP
FILE   HPNULL.OBJ
FILE   HINULL.OBJ
FILE   PSNULL.OBJ
FILE   MAIN.OBJ
FILE   BORDER.OBJ
FILE   CIRCLE.OBJ
FILE   DIMLIN.OBJ
FILE   FOOTS.OBJ
```

References.

Microsoft. (1989A). *Microsoft BASIC: programmer's guide (version 7.1)*, Microsoft Corporation, Redmond, WA.

________. (1989B). *Microsoft BASIC: BASIC language reference (version 7.1)*, Microsoft Corporation, Redmond, WA.

Ryan-McFarland. (1987). *RM/FORTRAN (version 2.4)*, Ryan-McFarland Corporation, Austin, TX.

Plotworks. (1989). *PLOT88: Software library and reference manual*, Plotworks, Incorporated, Ramona, CA.

Appendix A
Program Users Guide

The Camouflage Program is a collection of routines which are intended to allow the input of a structure description and desired materials, the modification of the data for an existing project, or execute an existing project. The contents of this appendix provide the instructions to install the Camouflage Program on any computer system and prepare that computer system for the execution of the Camouflage Program.

Program Installation. All of the necessary program modules and data files required for execution of the Camouflage Program are contained on DISK 1. A subdirectory should be created (MAKEd) on the hard disk of the computer system, and the contents of DISK 1 COPYed from the floppy disk to that subdirectory. A form of the system commands that will ‛c-complish these tasks are provided below:

```
MD \subdirname          creates subdirectory     .
CD \subdirname          changes to that subdirectory
COPY A:*.*              copies contents floppy to hard disk
```

Two stand-alone programs must be executed before the Camouflage Program can be executed. These two programs are executed only one time to define the existing computer system components for the Camouflage Program. The two programs are VALID and DEVICE (see Chapter 3 Utility Routines). If the computer system configuration is changed, or new components are added, these two programs should be executed again. The execution of these two programs will create two data files, COMFIL and DEVICES, on the resident (hard) disk whose contents are required for the Camouflage Program operation.

VALID Program. The VALID program is executed by typing the following:

```
VALID  <ENTER>
```

at the system level. An input screen with five data fields will be displayed, as follows:

```
U. S. Army Corps of Engineers
Waterways Experiment Station
Vicksburg, MS 39180

Camouflage Design Program Validation

PROGRAM RENEWAL DATE          12/31/1992

USER NAME

PROGRAM REGISTRATION NUMBER

SCRATCH DISK DRIVE

PROGRAM VERSION NUMBER        1.01

Press  <ESC>  when input is complete . . .
```

The first data field (PROGRAM RENEWAL DATE) will be highlighted. Movement to any of the other data fields is accomplished by the carriage return (<ENTER>) key, the directional arrow keys, the home (<HOME>) key, the end (<END>) key, the page up (<PGUP>) key or the page down (<PGDN>) key. When the input has been completed, the escape (<ESC>) key is used to exit the screen. The program will check the input data, and if they are satisfactory, the random access data file COMFIL is written, and the program is terminated. Each of the five data fields are described, and if any program restrictions exist for the data field, those restrictions are discussed as well.

PROGRAM RENEWAL DATE. The program renewal date is defaulted to 12/31/1992 by the program. The current version of the Camouflage Program does not use this data item; therefore, it need not be changed. If it is changed however, the format of a two digit month followed by a slash (/), two-digit day followed by a slash, and a four-digit year must be used. The program will reject any other format.

USER NAME. A user name of up to 25 characters may be provided. This data item is not checked by the program, and any combination of characters may be used. The current version of the Camouflage Program does not use this data item.

PROGRAM REGISTRATION NUMBER. Any number may be input. This data item is displayed on each page of program output and may be used to identify the computer which generated the output. The program checks to ensure that this number is between 1 and 99999. If the Camouflage Program resides on a single computer system, the input should be 1.

SCRATCH DISK DRIVE. This data item must be specified. Any legitimate disk drive specification may be provided. **A floppy disk drive**

should not be specified. If the computer system has a RAM drive available (D or E, for example), it would be advisable to use this disk drive because of its speed. If a RAM drive is specified, the system AUTOEXEC.BAT file should be modified to copy the Camouflage Program pricing data file (PRICE.LAT) and the three addressing data files (DATA.ADD, DATA.LOC, and DATA.FLD) from the hard drive to the RAM drive. The Camouflage Program will not transfer these data files, and assumes that they reside on the specified disk drive. If a subsequent execution of the Camouflage Program results in either an "File not found" error (System Error # 53) or an "Illegal function call" error (System Error # 5), one (or all) of these four data files is not resident on the assumed disk drive. These files may be copied to the designated disk drive with the COPY command, as follows:

```
COPY C:\CAMFLAGE\PRICE.LAT  D:
COPY C:\CAMFLAGE\DATA.*  D:
```

PROGRAM VERSION NUMBER. This data item is the number of the current version of the Camouflage Program. This data item may not be changed, and will be displayed on each page of program output.

When the data input are completed, the <ESC> key is used to terminate the input and initiate the data checking phase of the program. If any inconsistencies are detected, an appropriate message is displayed and control is returned to the input. When the data are assumed to be acceptable, the random access data file COMFIL is written and the program is terminated. No other executions of the VALID program are required by the Camouflage Program. The source codes for the VALID program are provided in Chapter 3 Utility Routines.

DEVICE Program. The DEVICE program is executed by typing the following:

```
DEVICE  <ENTER>
```

at the system level. Two input screens are displayed which allow the computer system monitor, printer and desired printer page size to be defined. The first input screen contains a list of available monitor types, as follows:

```
                    U. S. ARMY CORPS OF ENGINEERS
                   CAMOUFLAGE DESIGN OUTPUT DEVICES

DISPLAY ADAPTER          (Y/N)

TEXT MONITOR ONLY                Y
HERCULES MONO (720 BY 348)       N
CGA ( 2 COLOR 640 BY 200)        N
EGA (16 COLOR 320 BY 200)        N
EGA ( 2 COLOR 640 BY 200)        N
EGA (16 COLOR 640 BY 200)        N
EGA (MONOCHRM 640 BY 350)        N
EGA (16 COLOR 320 BY 200)        N
EGA (16 COLOR 640 BY 350)        N
VGA (2 COLOR 640 BY 480)         N
VGA (16 COLOR 640 BY 480)        N
VGA (256 COLOR 320 BY 200)       N

Press  <ESC>  when input is complete . . .
```

The first data field (TEXT MONITOR ONLY) will be highlighted. If another type of display adapter is being used, the Y for the TEXT MONITOR ONLY should be changed to an N and the highlighted field moved to the desired display adapter. The display adapter description which describes the monitor for the computer system should be indicated with a Y (y). This input may be terminated by the <ESC> key. When the <ESC> key is depressed, the indicated display adapter data field will flash for verification. If the flashing indicator is not the desired display, that data field should be changed to N and the desired data field changed to Y. When the desired display adapter data field is flashing, a second <ESC> keystroke will cause the screen to be cleared and the second input screen to be displayed, as follows:

```
                    U. S. ARMY CORPS OF ENGINEERS
                  CAMOUFLAGE DESIGN OUTPUT DEVICES

PRINTING DEVICE (Y/N)                    PAGE SIZE              (Y/N)

NO PRINTER                Y               8.5 BY 11.0 (A)        Y
EPSON (MX,RX-80)          N              11.0 BY 14.0 (AA)       N
EPSON (FX,JX-80)          N              11.0 BY 17.0 (B)        N
EPSON (FX-85)             N              17.0 BY 22.0 (C)        N
EPSON (FX-185,286)        N              22.0 BY 34.0 (D)        N
EPSON (FX-100)            N
EPSON (MX,RX-100)         N
EPSON (LQ-1500)           N
IBM (PRPRNTR/GRPHCS)      N
CENTRONICS                N
OKIDATA (92,182,192)      N
OKIDATA (93,193)          N
HP THINKJET               N
HP QUIETJET               N
HP QUIETJET PLUS          N
HP LASERJET               N

Press  <ESC>  to complete input or  <F1>  for previous screen ...
```

The first data field (NO PRINTER) will be highlighted. If another type of printer device is being used, the Y for the NO PRINTER should be changed to an N and the highlighted field moved to the desired printer device. The printer device description which describes the printer for the computer system should be indicated with a Y (y). The printer page size is specified similarly. If an 8.5- by 11.0-in. size is desired, no changes are required. If an 11.0- by 14.0-in. (wide carriage) page size is desired, the highlighted field should be moved to that data field and a Y provided (8.5 by 11.0 must be changed to N). When the <ESC> key is depressed, the indicated printer device and page size data fields will flash for verification. If the flashing indicators are not the desired device or page size, the data field(s) should be changed to N and the desired data field changed to Y. When the desired printer device and page size data fields are flashing, a second <ESC> keystroke will cause the screen to be cleared. When the data are assumed to be acceptable, the sequential data file DEVICES is written and the program is terminated. No other executions of the DEVICE program are required by the Camouflage Program. The source codes for the DEVICE program are provided in Chapter 3 Utility Routines.

Camouflage Program Execution. The Camouflage Program is executed by typing the following:

```
MAIN   <ENTER>
```

at the system level. A title screen will be displayed followed by the program options input, as follows:

```
Waterways Experiment Station
Environmental Engineering Division
Vicksburg, MS 39180

CAMOUFLAGE STRUCTURAL DESIGN
Program version dtd: May 15, 1991

OPTIONS: INPUT, EDIT, EXECUTE OR END (IN/ED/EX/END)?
```

Three program options are involved with any camouflage system. These program options provided are as follows:

(1) INPUT the data required to describe the soil conditions, system geometry, and desired materials for the project,

(2) EDIT the data for a project which has been previously defined, and

(3) EXECUTE the analysis and output the results for a project which has been previously defined.

The desired program option is indicated with a two-character keyword: IN, ED, EX, or EN (in, ed, ex, or en). If any other response is provided, an appropriate message is displayed and the input is corrected. If the program is not terminated (ENd), the next inputs required are the name and revision number of the project, as follows:

```
PROJECT NAME  (SIX CHARACTER MAX) ?

REVISION NUMBER  (0 - 9) ?
```

The project name can be any combination of letters and/or numbers up to six characters in length. If the project name is less than six characters long, the trailing blanks are filled with zeros. The revision number must be a numeric from zero to nine (0, 1, 2, 3, 4, 5, 6, 7, 8, or 9). The project name and revision number will be displayed on each page of program output, and is used to define the name of the project data file in which the data are stored (see Chapter 1 MAIN Program).

INput Program Option. The INput program option allows for the complete description of a project **which has not been previously defined.** If the input project name and revision number define an existing project data file, the program option is changed to EDit and the program operations are continued. This is done to avoid the potential disaster of overwriting the data for an existing project. Ten input screens will be displayed sequentially during the input process (see Chapter 5 CAMINP Routine). The input to any of the screens is terminated with either an <ESC> key or a <F1> key. Terminating with an <ESC> key will cause the next sequential input screen to be displayed, and terminating with a <F1> key will cause the previous input screen to be displayed. The use of these two terminating keys allows the user to move backwards through the screens (<F1>) and review the input to ensure that inconsistencies in the data do not occur. When an input screen is terminated with an <ESC>, limited checks on the data are made to ensure that the data are reasonable. If inconsistencies are detected, an appropriate message is displayed, and the user is required to correct the input before continuing to the next screen. Terminating an input screen with <F1> provides no such data checks. Each of the input screens is provided with a description of the data fields and data checks that are made. As each input screen is displayed, the first data field will be highlighted. Movement of the highlighted field is accomplished with the following keys (see CURSOR Routine in Chapter 3 Utility Routines): (a) <ENTER> key, (b) arrow keys, (c) <Home> key, (d) <End> key, (e) <Page Up> key, or (f) <Page Down> key. All other keys (except <ESC> and <F1> keys) are accepted as input or ignored.

The initial screen is the general site description input screen as follows:

```
            FIXED FACILITY SUPPORT SYSTEMS     PROJECT # TRIAL
                  Site Specific Data          REVISION # 0

       SOILS INVESTIGATION and FOUNDATION DATA

             cohesion (psf):  400.
             unit weight (pcf):  120.
             angle of internal friction (degrees):  25.0
             diameter of boring for pole socket (inches):  12.0

       CLIMATE DATA

                 average daily rainfall (inches):   0.10
                 average daily temperature (degs F):   50.0
                 wind speeds (mph):  10.0
                      sustained average (mph):
                      gust speeds (mph):
                 potential ice or snow load (psf): 2.0

       GENERAL TERRAIN AND FOLIAGE CLASSIFICATION:
```

There are 11 data items shown on the Site Specific Data input screen. If any data item has a program default value, that value is also displayed. If no program default value exists for a data item, the data field is blank. The program default values are provided on the input screen shown, and are also listed in Chapter 5 CAMINP Routine. These values may be changed by the user. When the <ESC> key is depressed, the input values for the soil cohesion, soil unit weight, and soil angle of internal friction are checked to ensure that reasonable values are provided. No other data items are checked.

The next input screens involves the description of the structure geometry, as follows:

```
             FIXED FACILITY SUPPORT SYSTEMS      PROJECT # TRIAL
                   Structure Geometry          REVISION # 0

       PLAN VIEW DATA

            length of structure (feet):
            bay spacings (measured along the structure length)

                  @         @         @         @         @

                  @         @         @         @         @

             width of structure (feet):
             minimum pole spacing (feet): 10.0

       ELEVATION DATA

             exterior pole height (above ground feet): 20.0
             exterior poles guyed (Y/N): Y
             allowable netting sag (feet): 2.0
             allowable displacement error (inches):  0.10
```

There are 27 data items shown on the Structure Geometry input screen. If any data item has a program default value, that value is also displayed. If no program default value exists for a data item, the data field is blank. The program default values are provided on the input screen shown, and are also listed in Chapter 5 CAMINP Routine. These values may be changed by the user. When the <ESC> key is depressed, the input values for the structure length, structure width, minimum support spacing along the width, support height, maximum sag in the tension members, and maximum displacement error are checked to ensure that none were left blank. In addition, the specified bay spacings are checked to ensure that the sum of the bay spacings is equal to the structure length.

The remaining eight input screens provide for a description of the structural materials (support members, netting, tension members, and anchors) to be used for the project. Each structural material item provides two screens on which up to 30 material items may be provided. Minimum input required for each structural material is the complete definition of each data item listed for one (1) material item. The first material input screen(s) provide for the description of the desired support member(s), as follows:

```
        FIXED FACILITY SUPPORT SYSTEMS      PROJECT # TRIAL
        Support Member Material Screen     REVISION # 0

INDICATE DESIRED SUPPORT MEMBER TO BE USED  (Y-Yes  N-No)

                    ALLOW    MODULUS X-SECTION  MOMENT   UNIT
                    LENGTH  STRENGTH  ELASTIC   AREA    INERTIA   PRICE
   MARK  (FT)       (KSI)    (10**6)   (SQIN)   (IN**4)  ($/FT)  DESCRIPTION
 N A0202P  20        30       10        1.75     .911     0        2 X 2 ALUMINUM
 N A0303P  20        30       10        2.75     3.49     0        3 X 3 ALUMINUM
 N A0404P  20        50       29.0      6.36     12.35    0.80     4 X 4 ALUMINUM
 N W0404P  20         7       1.7       16       85       0        4 X 4 WOODEN PO
```

Nine data fields are provided for each support member. Before this input screen is displayed, the project pricing data file (PRICE.LAT) will be searched for up to 30 data items having a use code of P (see Chapter 5 CAMINP Routine and Chapter 4 Pricing Data). Each material item extracted (maximum of 15 per screen) will be displayed with its characteristics as shown. If the material characteristics for a particular item are not defined in the PRICE.LAT data file, values of 0 will be displayed. If no material items with a use code of P are found, the screen will be blank. **Only one support member may be selected.** One material item must be provided and completely described as to the characteristics shown before an <ESC> key will be accepted. Only the data for the selected material item are checked. The first data field for each material item is a Y or N to indicate whether the material is to be used or not. The MARK of the material item is the five-character name of the material and the one-character use code. The name and the 15-character description of the selected material will be displayed in the material summary output. The remaining six data items must be specified (non-zero and non-blank) before execution is allowed to continue. If more than 15 material items are provided, the user may <ESC> to the second support member input screen and <F1> back to the first support member input screen to ensure that the correct member has been selected. An <ESC> on the second support member input screen initiates the data checks.

The second material input screens involve the desired netting, as follows:

```
                         FIXED FACILITY SUPPORT SYSTEMS     PROJECT # TRIAL
                             Netting Material Screen       REVISION # 0

 INDICATE DESIRED NETTING TYPE TO BE USED  (Y-Yes  N-No)

          UNIT  NETTING  ALLOW     UNIT     DRAG     UNIT
          SIZE   THICK  STRENGTH  WEIGHT   COEFFIC   PRICE
   MARK  (SQFT)  (IN)    (KSI)   (#/SQFT)  (UNITS) ($/SQFT)   DESCRIPTION
N BF016N   0      0        0        0         0        0      BRUNSWICK 16-LB
N BF022N   0      0        0        0         0        0      BRUNSWICK 22-LB
N SF050N 400.0  0.085     5.0     0.083     .005     0.15     STANDARD DOD50
N TB035N   0      0        0        0         0        0      TELEDYNE BROWN
N UL060N   0      0        0        0         0        0      ULCAN 60-LBNET
```

Nine data fields are provided for each net. Before this input screen is displayed, the project pricing data file (PRICE.LAT) will be searched for up to 30 data items having a use code of N (see Chapter 5 CAMINP Routine and Chapter 4 Pricing Data). Each material item extracted (maximum of 15 per screen) will be displayed with its characteristics as shown. If the material characteristics for a particular item are not defined in the PRICE.LAT data file, values of 0 will be displayed. If no material items with a use code of N are found, the screen will be blank. **Only one netting material may be selected.** One material item must be provided and completely described as to the characteristics shown before an <ESC> key will be accepted. Only the data for the selected material item are checked. The first data field for each material item is a Y or N to indicate whether the material is to be used or not. The MARK of the material item is the five-character name of the material and the one-character use code. The name and the 15-character description of the selected material will be displayed in the material summary output. The remaining six data items must be specified (non-zero and non-blank) before execution is allowed to continue. If more than 15 material items are provided, the user may <ESC> to the second netting input screen and <F1> back to the first netting input screen to ensure that the correct item has been selected. An <ESC> on the second netting input screen initiates the data checks.

The third material input screens involve the desired tension member, as follows:

```
                                   FIXED FACILITY SUPPORT SYSTEMS     PROJECT # TRIAL
                                   Tension Member Material Screen    REVISION # 0

 INDICATE DESIRED TENSION MEMBER TO BE USED  (Y-Yes  N-No)

            ALLOW    MODULUS  THERMAL X-SECTION  UNIT    UNIT
          STRENGTH  ELASTIC   COEFFIC   AREA    WEIGHT   PRICE
    MARK   (KIPS)   (10**6)  (10**-6)  (SQIN)   (#/FT)  ($/FT)    DESCRIPTION
N GR416C     0        0         0        0        0       0     1/4 INCH GRASS
N GS216C   16.0     29.0       6.5     .012      0.40     0     1/8 IN AIRCRAFT
N GS316C   17.5     29.0       6.5     .785      0.45    0.70   3/16 IN AIRCRAF
N GS416C   20.0      0          0        0        0       0     1/4 IN AIRCRAFT
N GS516C   22.0      0          0        0        0       0     5/16 IN AIRCRAF
N GS616C     0       0          0        0        0       0     3/8 IN AIRCRAFT
N GS816C     0       0          0        0        0       0     1/2 IN AIRCRAFT
N NL416C     0       0          0        0        0       0     1/4 INCH NYLON
```

Nine data fields are provided for each tension member. Before this input screen is displayed, the project pricing data file (PRICE.LAT) will be searched for up to 30 data items having a use code of C (see Chapter 5 CAMINP Routine and Chapter 4 Pricing Data). Each material item extracted (maximum of 15 per screen) will be displayed with its characteristics as shown. If the material characteristics for a particular item are not defined in the PRICE.LAT data file, values of 0 will be displayed. If no material items with a use code of C are found, the screen will be blank. **Only one tension member material may be selected.** One material item must be provided and completely described as to the characteristics shown before an <ESC> key will be accepted. Only the data for the selected material item are checked. The first data field for each material item is a Y or N to indicate whether the material is to be used or not. The MARK of the material item is the five-character name of the material and the one-character use code. The name and the 15-character description of the selected material will be displayed in the material summary output. The remaining six data items must be specified (non-zero and non-blank) before execution is allowed to continue. If more than 15 material items are provided, the user may <ESC> to the second tension member input screen and <F1> back to the first tension member input screen to ensure that the correct item has been selected. An <ESC> on the second tension member input screen initiates the data checks.

The final material input screens involve the desired anchors for a guyed structural system, as follows:

```
                         FIXED FACILITY SUPPORT SYSTEMS     PROJECT # TRIAL
                            Anchor Material Screen        REVISION # 0

 INDICATE DESIRED ANCHOR TO BE USED  (Y-Yes  N-No)

                     ALLOW    MODULUS    ROD     ANCHOR   UNIT
             LENGTH STRENGTH  ELASTIC DIAMETER DIAMETER  PRICE
   MARK      (INCH)  (KIPS)   (10**6)  (INCH)   (INCH)   ($/EA)     DESCRIPTION
N T0146A      66       58       29      1.25      10      32.8    1 1/4 X 66 10-I
N T0148A      96       58       29      1.25      10      39.6    1 1/4 X 96 10-I
N T0816A      66       36       29      1.00     8.0      20.5    1 X 66 8-INCH T
N T2537A      96       58       29      1.25      14      59.6    1 1/4 X 96 14-I
N T4345A      54       23       29       .75       4       11     3/4 X 54 4-INCH
N T6346A      66       23       29       .75       6      13.7    3/4 X 66 6-INCH
```

Nine data fields are provided for each anchor. Before this input screen is displayed, the project pricing data file (PRICE.LAT) will be searched for up to 30 data items having a use code of A (see Chapter 5 CAMINP Routine and Chapter 4 Pricing Data). Each material item extracted (maximum of 15 per screen) will be displayed with its characteristics as shown. If the material characteristics for a particular item are not defined in the PRICE.LAT data file, values of 0 will be displayed. If no material items with a use code of A are found, the screen will be blank. **Up to five (5) anchor materials may be selected.** These selected material items must be provided and completely described as to the characteristics shown before an <ESC> key will be accepted. Only the data for the selected material items are checked. The first data field for each material item is a Y or N to indicate whether the material is to be used or not. The MARK of the material item is the five-character name of the material and the one-character use code. The name and the 15-character description of the selected material will be displayed in the material summary output. The remaining six data items must be specified (non-zero and non-blank) before execution is allowed to continue. If more than 15 material items are provided, the user may <ESC> to the second anchor input screen and <F1> back to the first anchor input screen to ensure that the correct items have been selected. An <ESC> on the second anchor input screen initiates the data checks.

At the conclusion of the input, the project data are copied from the designated scratch disk drive to the floppy disk drive. **A disk must reside in the floppy disk drive while the program is executing.** When the data have been transferred to the floppy disk, the input program function is complete, and control is returned to the original program option prompt.

EDit Program Option. The EDit program option allows for the review and/or modification of the data for a project **which has been previously defined.** If the input project name and revision number do not define an existing project data file, an error message is displayed and the PROJECT NAME prompt is displayed. If a list of the defined project data file names contained on the floppy disk are desired, a response of DIR will produce such a list (see OUTDIR Routine in Chapter 3 Utility Routines). When an existing project name and revision number are provided, the data from the corresponding project data file are transferred from the floppy disk to the designated scratch disk drive. The ten input screens described in the INput Program Option section above are displayed sequentially during the edit process (see Chapter 5 CAMINP Routine). Each input screen is terminated with an <ESC> or an <F1> key, and the data checks are made with an <ESC> termination as described above. When the last input screen has been reviewed/modified successfully, the data are transferred to the floppy disk drive, and control is returned to the original program option prompt.

EXecution Program Option. The EXecution program option initiates the design for the structure as defined in the specified project data file. Designs are initiated for the overall structure (see Chapter 6 DESIGN Routine), the footing/foundations required (see Chapter 7 FOOTING Routine), and the anchoring system (if the system is guyed, see Chapter 8 ANCDES Routine). As the individual designs are being accomplished, an appropriate display is provided on the screen. At the conclusion of the system designs, the project output is begun (see Chapter 9 Output Routines). The initial output form is a plan view of the final structure. The plan view is displayed on the monitor screen. The plan view can be cleared from the screen by depressing any key, and the following prompt will be displayed:

```
Plot to Printer     (Y/N)?
```

If a printer plot is desired, a response of Y (y) should be provided. The plan view displayed originally on the screen will be transmitted to the printer. The remaining output forms will be generated to a designated sequential data file, and at the conclusion of this output, the following prompt will be displayed:

```
File Name:  Pproject0.PRT

                 U T I L I T Y   M E N U
                 ________________________


                 B - Browse the file
                 C - Copy the file to drive A:
                 P - Print the file
                 Q - Quit

                 Enter Selection (B/C/P/Q) -
```

The B(rowse) option will display the designated output file on the screen and the user may review the output. The arrow, <Home>, <End>, <Page Up>, and <Page Down> keys may be used to scroll through the output.

The C(opy) option will copy the designated output file from the resident (hard) disk drive to a floppy disk on the A disk drive.

The P(rint) option will output the designated output file to the printer.

The Q(uit) option will allow the user to save the designated output file on the resident disk drive or erase the output file. It is recommended that if the output is to be retained, copy the output to the same floppy disk containing the project data and erase the output file from the hard disk drive.

EDITPRIC Program. The pricing data file (PRICE.LAT) is an integral program element during the INput program process. The contents of this data file are searched before the support member, netting, tension member, and anchoring input screens are displayed. As new materials become available, or when existing material data items need updating, the pricing data file can be used to maintain a permanent record of these data. If a user were to simply edit the project data input screens, those changes would affect only that project and would not affect any other project. As the data for any new project are being input, the current contents of the pricing data file are searched, and updated/added materials are included in the material screens. The EDITPRIC program may be used to update existing material characteristics, add new materials to the pricing data file, or delete existing material items from the pricing data file. The EDITPRIC program is executed at the system level by typing the following:

```
EDITPRIC  <ENTER>
```

The stand-alone executable program **EDITPRIC** has been provided in the library of program modules, and is to be used to add material names and associated data to the contents of the pricing data file. The following conventions must be adhered to while adding data to the pricing data file:

(1) Existing material MARK names should not be duplicated; if the MARK name exists in the current version of the PRICE.LAT data file, subsequent data inputs will replace the current data in the data file. The first five characters of the MARK name can be any combination of letters and/or numbers or symbols, but the user is encouraged to use a systematic naming convention to avoid later confusion. The sixth character of the MARK name will be the use code of the data item.

(2) The use code for the material item to be added must follow the convention of the current list of use codes incorporated in the program (see Table A1 below). If a new use code is to be defined, a record of this use code must be maintained, so that when similar materials (with the same use code) are added, the same use code will be used. Simply stated: if an additional support member is to be included in the PRICE.LAT data file,

Table A1
Current Pricing Data File Use Codes and Descriptions

Use Code	Description
A	Anchors and related materials
B	Bolts and primary connectors
C	Structural tension members (cables)
E	Eyebolts
M	Pulleys
N	Netting
P	Structural support members
R	Rings and snap connectors
T	Turnbuckles

the use code for that item must be specified as 'P'. The same applies for the other items listed in Table A1.

(3) Careful attention must be provided as to the required units of each data item. The current version of the Camouflage Program assumes that the convention unit of distance is provided in feet, area in square feet, material strengths in either kips or kips per square inch and so on.

A brief discussion of the operation of the **EDITPRIC** program is provided to illustrate the required input conventions described above. To execute the program, simply type the following:

```
EDITPRIC  <ENTER or RETURN>
```

A title screen will appear followed by the program options prompt, as shown below:

```
INPUT, EDIT, DELETE, DISPLAY, OR END (IN/ED/DE/DI/END) ?
```

One of the five two-character program options should be provided. The program options allow for the INput of a single material item and its data, the EDiting of a single material item's data, for the DEletion of a single material item and its data, the DIsplay of the use code currently being used, and for the program to be ENDed.

Input Program Option. A response of IN (in) will allow the INput of one material item with its associated data. A prompt requiring the six-character MARK INDEX (name) of the item to be input will appear. Once this input is provided, the screen shown below will be displayed, and the user should input the data for the other seventeen data fields. The input will be terminated when the <ESC> key is depressed. An example of this input is provided as follows:

```
INPUT, EDIT, DELETE, DISPLAY, OR END (IN/ED/DE/DI/END) ? in

Input Item Mark Index or END?  ABCXYZ

                 U. S. ARMY CORPS OF ENGINEERS
                        UNIT PRICING DATA

RECORD NO.   158
                 ITEM MARK INDEX                        ABCXYZ
                 PRIMARY CATEGORY
                 MATERIAL SIZE
                 PRICING UNIT OF MEASURE
                 ALLOWABLE STRENGTH (KSI)
                 ESTIMATE UNIT OF MEASURE
                 ESTIMATE UNIT WEIGHT (WET OR DRY)
                 FEDERAL STOCK NUMBER
                 FABRICATED PART/USE DESCRIPTION
                 MODULUS OF ELASTICITY (10**6)
                 X-SECT AREA/DIAMETER/THICKNESS
                 MOMENT OF INERTIA
                 COEFF OF THERMAL EXPANSION (10**-6)
                 ANCHOR SIZE (HELIX, INCHES)
                 ITEM UNIT PRICE
                 ERECTION TIME (MAN-HRS)
                 DRAG COEFFICIENT (NET ONLY)
                 NOT IN USE CURRENTLY

                 Press  <ESC>  when input is complete . . .
```

Edit Program Option. The EDit program prompt allows the data for the specified item to be modified and restored to the current pricing data file. If the specified item MARK does not currently exist in the pricing data file, an error message is displayed and the original prompt repeated. If the specified MARK is found in the data file, the values of the individual data items will be displayed, and the user may change any of the seventeen values (excluding the MARK name of the item). The current contents of the pricing data will be modified at the conclusion of each EDit program function.

Delete Program Option. The DElete program function will allow a specified MARK name and its data to be deleted from the pricing data file. This program function should only be used if the current pricing data file is full (2,002 data items), and any unused data items are to be deleted to provide the space for new data items. Extreme care should be taken when using this program function.

Display Program Option. The DIsplay program function will provide a display of the current contents of the pricing data file. This display will provide the number of data items and the beginning record numbers currently existing for each of the 37 use codes. No other user input is allowed, and the display is erased when any key is depressed. An example of this display is provided below:

```
                                PRICE.LAT  CONTENTS

 NUM  CHAR  BEGAD  NOITEM  NUM  CHAR  BEGAD  NOITEM  NUM  CHAR  BEGAD  NOITEM
   1            0       0   13    B       7      11   25    N      45       5
   2    0       0       0   14    C      18       8   26    O      50       0
   3    1       0       0   15    D      26       0   27    P      50       4
   4    2       0       0   16    E      26      13   28    Q      54       0
   5    3       0       0   17    F      39       0   29    R      54       6
   6    4       0       0   18    G      39       0   30    S      60       0
   7    5       0       0   19    H      39       0   31    T      60      19
   8    6       0       0   20    I      39       0   32    U      79       0
   9    7       0       0   21    J      39       0   33    V      79       0
  10    8       0       0   22    K      39       0   34    W      79       0
  11    9       0       0   23    L      39       0   35    X      79       0
  12    A       1       6   24    M      39       6   36    Y      79       0
                                                      37    Z      79       1
  NO. RECORDS      79
```

The data displayed in the table above provide the number of the use code (NUM), the associated one-character use code (CHAR), the beginning record number (BEGAD) of the use code, and the number of material items (NOITEM) in each use code. From these data, the use code A is the first defined use whose material data begin in the first available record (BEGAD=1) and there are six (NOITEM=6) material items with a use code of A. After the display table has been cleared from the screen, the following prompt will be provided:

```
LIST MARKS FOR USE CODE      OR END?
```

A one-character use code can be provided, and the marks (names) of any material items with that use code will be listed. The display function can be used to ensure that item marks are not duplicated.

End Program Option. When the desired program functions are completed, the program may be terminated by providing END (end) to the original program option prompt.

Appendix B
Design Routine Formulations

DESIGN INPUT. The input file for the design program must include the following information:

1) basic plan view dimensions,

2) pole sizes and material properties,

3) cable sizes and material properties,

4) structural loads, and

5) design coefficients.

The basic plan view dimensions include the number of bays and each bay width in one direction, and the total length perpendicular to these bay widths. The direction perpendicular to the bay widths is the design direction. The distance between support members in the design direction is determined through the design process. Support members in the design direction are equally spaced throughout the structure. All of these distances must be given in feet (see Figure B1).

All support members in the structure must be the same size and have the same material properties. Support member size must include height in feet (see Figure B2), moment of inertia in inches^4, cross-sectional area in square inches, cross-sectional width or diameter in inches, and the minimum allowed distance between members in feet. The minimum allowed distance between members allows the user to specify a distance which is considered too dense for practical use. Material properties for the support members must include modulus of elasticity and yield strength in pounds per square inch.

All cables in the structure must have the same material properties and cross-sectional area. Cross-sectional area of the cable must be given in square inches. A maximum sag distance must be defined which controls all cables. The maximum sag distance is the maximum distance in feet the cable may sag at its midpoint relative to its two ends (see Figure B3).

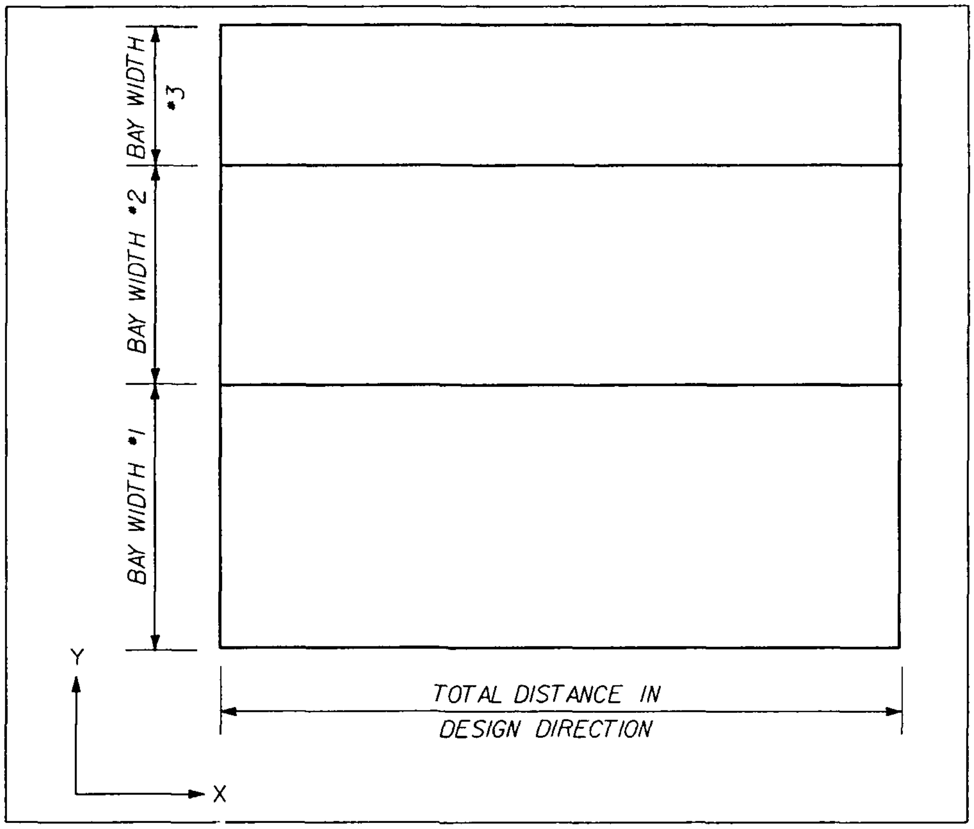

Figure B1. Basic plan view dimensions

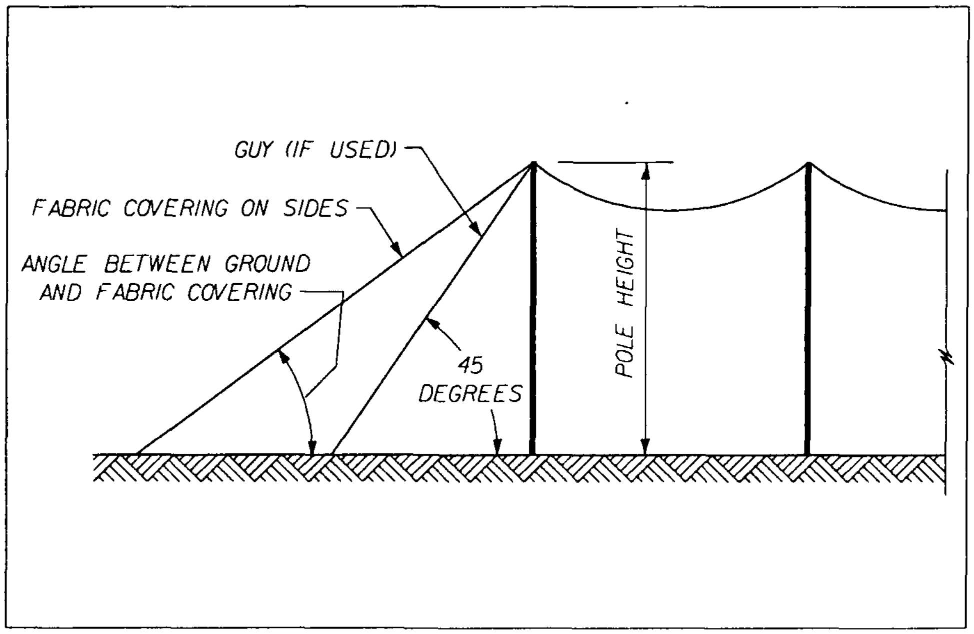

Figure B2. Support member characteristics

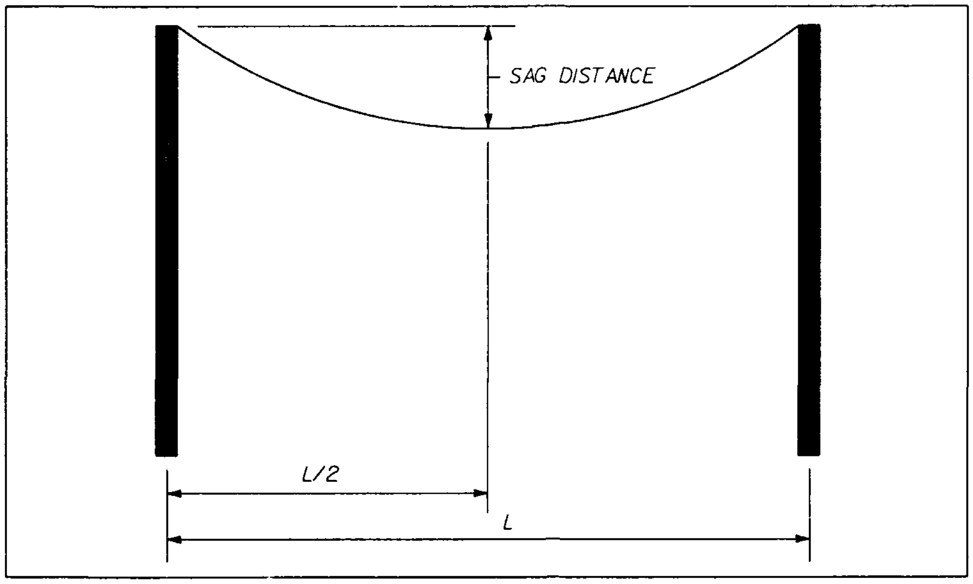

Figure B3. Tension member sag distance

The material properties of the cable must include the modulus of elasticity in pounds per square inch, the maximum tensile force the cable can maintain in pounds, and the linear weight of the cable in pounds per foot.

Loading information must include the netting or fabric covering wet weight and snow load in pounds per square foot, and the wind velocity in miles per hour.

Coefficients needed for the procedure include the drag coefficient for the fabric covering, and the factor of safety to be used for the entire structure. The structure must also be defined as being with or without guys on all exterior poles. If guyed, the guys are installed at a 45-degree angle connecting the top of the pole to the ground anchors (see Figure B2). The angle that the fabric covering makes with the ground on the sides of the structure must also be specified in degrees (see Figure B2). It is assumed that the fabric covering does not come in contact with the guys. The angle for the fabric covering must therefore be an angle less than 45 degrees.

NODE AND CABLE NUMBERING. For a given trial pole spacing in the design direction, the number of nodes and cables can be calculated.

$$\begin{array}{c}\text{Number of Poles in}\\ \text{Bay Width Direction}\,(NPY)\end{array} = \begin{array}{c}\text{Number of}\\ \text{Bays}\,(NB)\end{array} + 1 \qquad \text{(B1)}$$

$$\begin{array}{c}\text{Number of Poles in}\\ \text{Design Direction}\,(NPX)\end{array} = \frac{\begin{array}{c}\text{Total Length in}\\ \text{Design Direction}\,(TL)\end{array}}{\begin{array}{c}\text{Trial Pole Spacing in}\\ \text{Design Direction}\,(PSPC)\end{array}} + 1 \qquad \text{(B2)}$$

$$\text{Number of Total Poles } (NP) = NPX * NPY \tag{B3}$$

$$\begin{aligned}\text{Number of Horizontal Cables } (NC) &= 2 * [(NPX - 1) + (NPY - 1)] + 2 * (NPX - 1) \\ &* (NPY - 1) + (NPX\ 2) * (NPY - 1) \\ &+ (NPX - 1) * (NPY - 2)\end{aligned} \tag{B4}$$

In Equation B4, the first term counts the perimeter cables, the second term the diagonal cables, the third term the interior cables in the Y-direction (bay dimension), and the fourth term the interior cables in the X-direction (design dimension).

If the structure is guyed, the number of guys is given by the following equation.

$$\text{Number of Guys } (NG) = 2 * (NPX + NPY) \tag{B5}$$

Each cable is assigned a beginning node and an ending node. If the cable is parallel to the bay width or Y-direction, the beginning node is the lower node, and the ending node is the higher node. If the cable is parallel to the design or X-direction, the beginning node is the left node, and the ending node is the right node. If the cable is a diagonal sloping up and to the right, the beginning node is the lower left node, and the ending node is the upper right node. If the cable is a diagonal sloping up and to the left, the beginning node is the upper left node, and the ending node is the lower right node. All guys are defined as beginning at the only node to which they are attached and ending at an imaginary node number. Guys along the bottom line of the plan view have imaginary node number -1000. Guys along the top line of the plan view have imaginary node number -2000. Guys along the left line of the plan view have imaginary node number -3000. Guys along the right line of the plan view have imaginary node number -4000. The imaginary node numbers define the direction of the guys.

Each node is classified as one of nine types based on the number of cables and configuration of the cables which have that node as their beginning end. The node types are shown in Figure B4.

Node type (1) is node (1). Node type (1) is the beginning node for three horizontal cables and two guys. The three horizontal cables are numbered starting with the cable parallel to the Y-direction, then the diagonal cable sloping up and to the right, and ther the cable parallel to the X-direction (see Figure B5). If guys are present, all guys are numbered after all horizontal cables have been numbered.

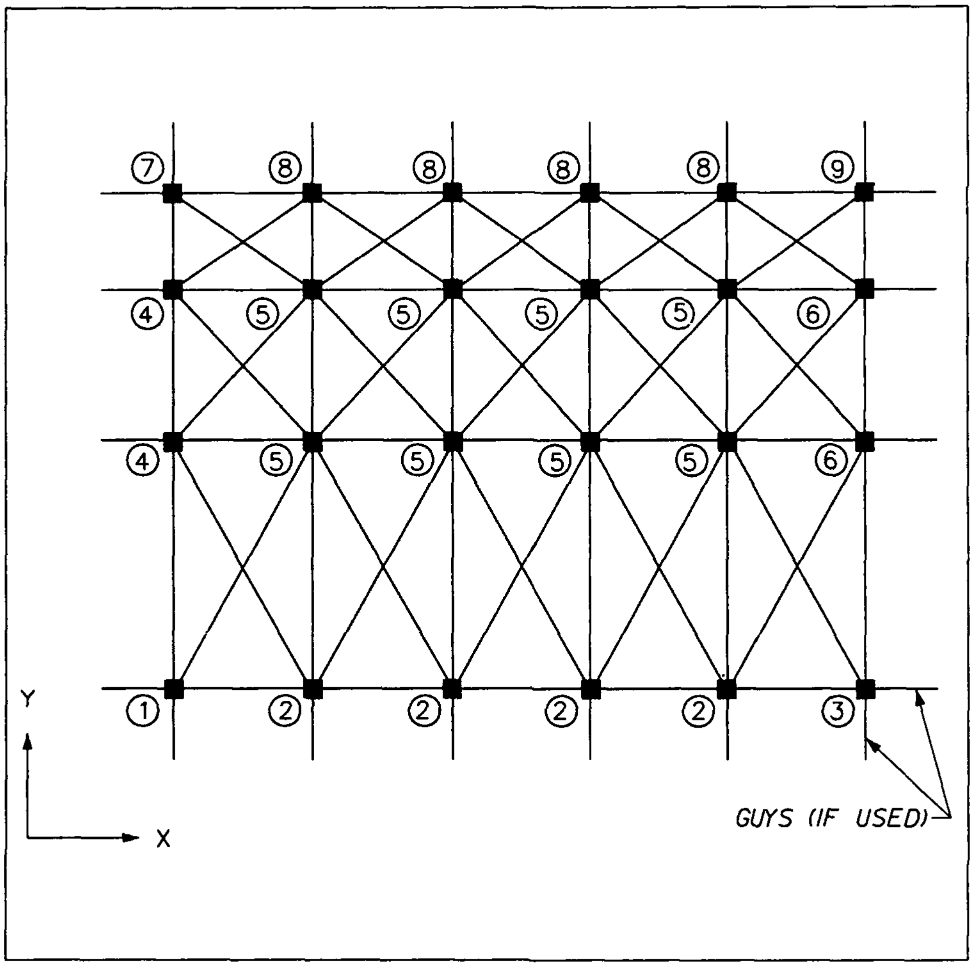

Figure B4. Node types

Node t, pe (2) is any node on the bottom line of the plan view excluding the two endpoints. Node type (2) is the beginning node for four horizontal cables and one guy. The four horizontal cables are numbered starting with the diagonal cable sloping up and to the left, then the cable parallel to the Y-direction, then the diagonal cable sloping up and to the right, and then the cable parallel to the X-direction.

Node type (3) is the right-most node on the bottom line of the plan view. Node type (3) is the beginning node for two horizontal cables and two guys. The two horizontal cables are numbered starting with the diagonal cable sloping up and to the left, and then the cable parallel to the Y-direction.

Node type (4) is any node on the left-most line of the plan view excluding the endpoints. Node type (4) is the beginning node for three horizontal cables and one guy. The three horizontal cables are numbered starting

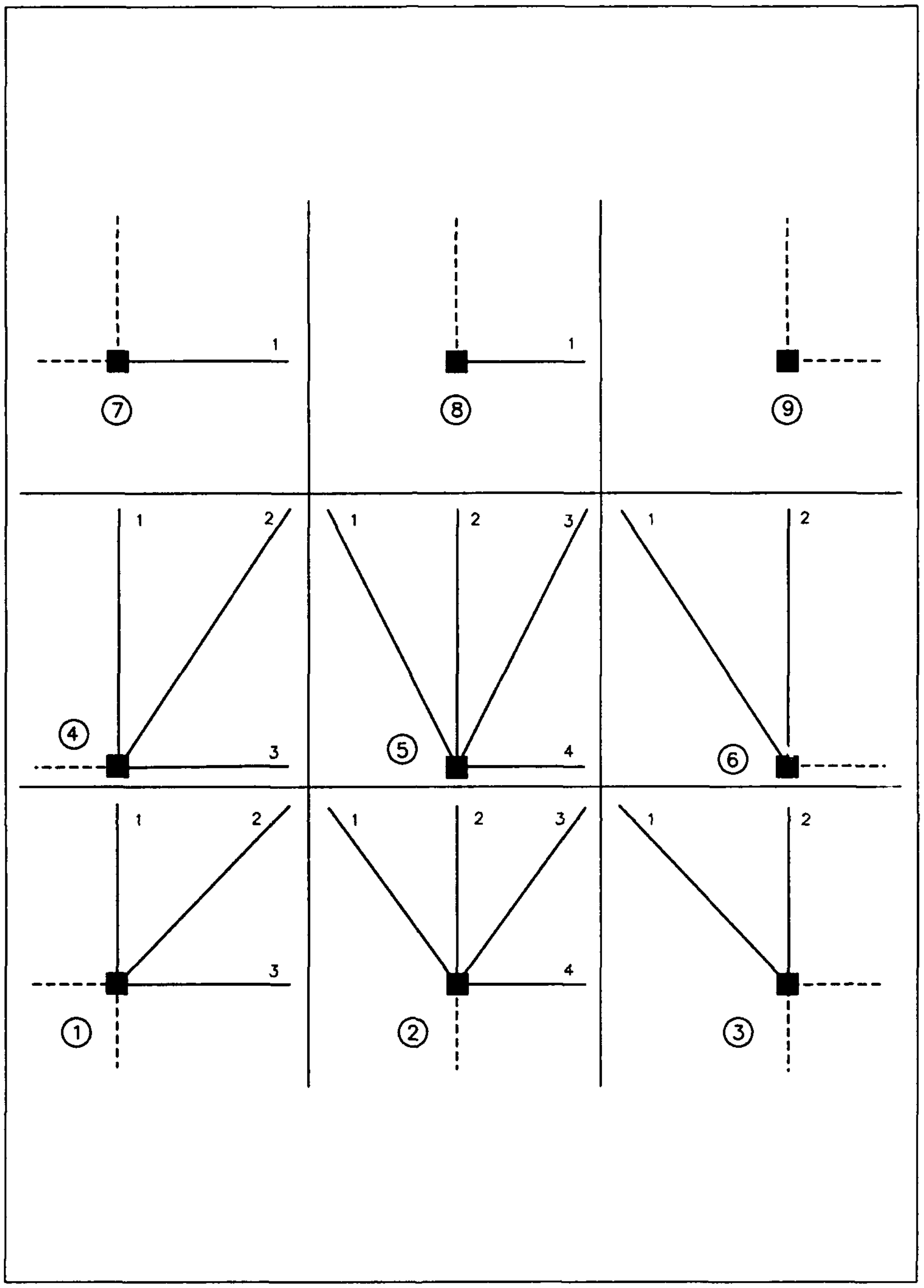

Figure B5. Cable numbering

with the cable parallel to the Y-direction, then the diagonal cable sloping up and to the right, and then the cable parallel to the X-direction.

Node type (5) is any node on the interior of the structure. Node type (5) is the beginning node for four horizontal cables and no guys. The four horizontal cables are numbered starting with the diagonal cable sloping up and to the left, then the cable parallel with the Y-direction, then the diagonal cable sloping up and to the right, and then the cable parallel to the X-direction.

Node type (6) is any node on the right most line of the plan view excluding the endpoints. Node (6) in the beginning node for two horizontal cables and one guy. The two horizontal cables are numbered starting with the diagonal cable sloping up and to the left, and then the cable parallel to the Y-direction.

Node type (7) is the left-most node on the top line of the plan view. Node type (7) is the beginning node for one horizontal cable and two guys. The horizontal cable is a cable parallel to the X-direction.

Node type (8) is any node on the top line of the plan view excluding the endpoints. Node type (8) is the beginning node for one horizontal cable and one guy. The horizontal cable is a cable parallel to the X-direction.

Node type (9) is the right-most node of the top line of the plan view. Node type (9) has no horizontal cables and two guys.

All horizontal cables are numbered before any guys are numbered if guys are present. The horizontal cables are numbered consecutively moving from node to node consecutively. All horizontal cables beginning at node (1) are numbered in the order of its node type, then the horizontal cables beginning at node (2) are numbered in the order of its node type, and so on until node by node and cable by cable all of the cables are numbered.

If guys are present, the guys are numbered beginning with the number following the final horizontal cable. All guys on the bottom line of the plan view are numbered from left to right, then all guys on the top line of the plan view are numbered from left to right, then all guys on the left-most line of the plan view are numbered from bottom to top, and then all guys on the right-most line of the plan view are numbered from bottom to top.

UNIFORM DISTRIBUTED LOADS FOR EACH CABLE TYPE. The applied loads of cable weight, fabric covering weight, and snow load are reduced to a single uniform distributed load along the sagged cable length as previously described. Each cell in the structure is divided into four triangles by the diagonal cables. Each one of these triangles is divided into three more triangles by passing lines from the three vertices to the centroid of the larger triangle as shown in Figure B6. The loads carried by each load triangle is applied to the cable adjacent to it.

Cable weight and fabric covering weight are considered to be distributed along the sag curve of the cables already, although not uniformly, but the snow load is given as a load per horizontal area. The fabric covering weight is taken as the wet weight of the fabric in order to obtain the

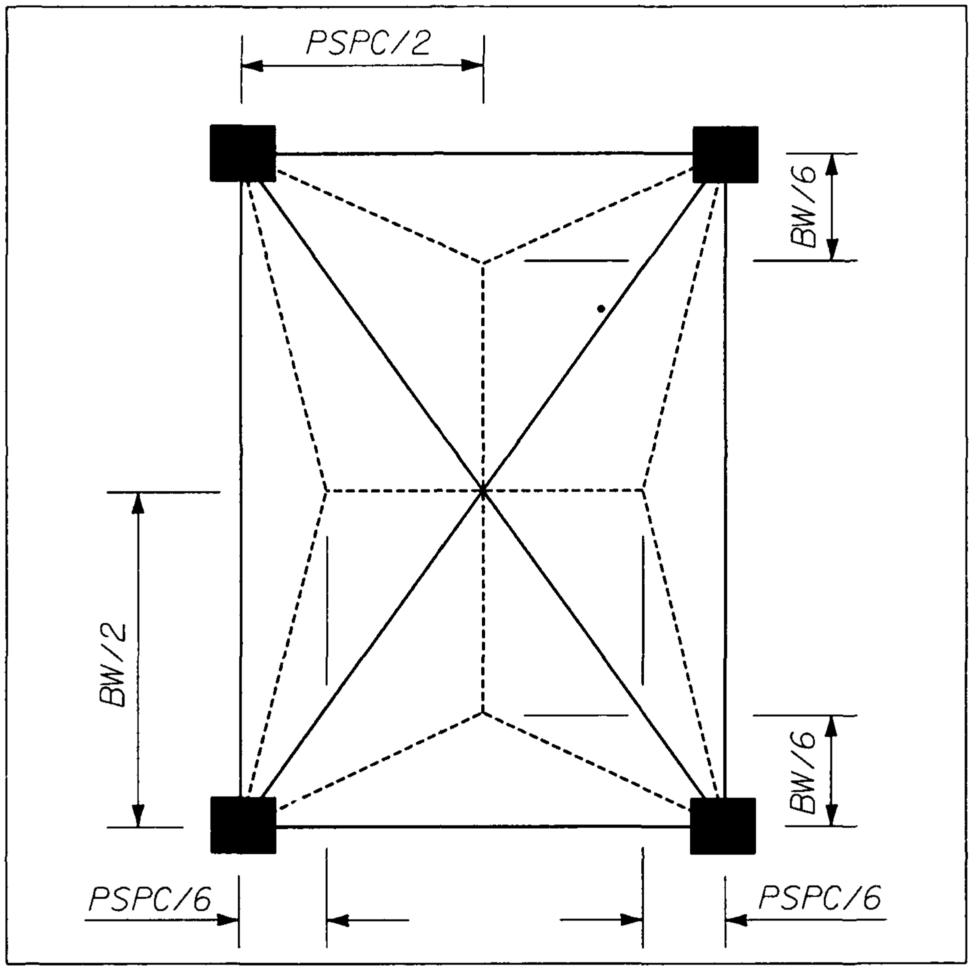

Figure B6. Single cell load triangles

largest load case. All loads are then reduced to a uniformly distributed load along the sag curve of the cable. Since the distributed load must be known before the actual length of the cable can be calculated, the given sag distance is added to the horizontal length of the cable to approximate the actual sagged length of each load triangle.

Cables are classified as one of five types: (1) a bay-length cable on the exterior of the structure, (2) a bay-length cable on the interior of the structure, (3) a trial-pole-space-length cable on the exterior of the structure, (4) a trial-pole-space-length cable on the interior of the structure, and (5) a diagonal cable. Cable type (1) carries the loads from the adjacent load triangle and the loads from the fabric and snow on the adjacent side panel of fabric. Cable type (2) carries the loads from the adjacent load triangles on each side. Since these triangles have the same area, the load on one load triangle can be calculated and doubled. Cable type (3) carries the loads from the adjacent load triangle and the loads from the fabric and snow on the adjacent side panel of fabric. Cable type (4) carries the loads from the adjacent load triangles on each side. These load triangles may not have the same area since bay widths may not be the same; therefore, each must be calculated and added to one another. Cable type (5) carries the loads from four adjacent load triangles. Each one of these load triangles has the same area, so one can be calculated and quadrupled. (See Figure B7.)

For cable type (1), the uniform distributed load due to snow load, wet fabric weight, and cable weight is:

$$q = \left[\frac{PH}{\tan\,(ANGLE)} + \frac{PSPC}{12}\right] * SNOW$$

$$+ \frac{[BW + SMAX] * CAMOWWT}{BW} * \left[\frac{PH}{\sin}\,(ANGLE) + \frac{13}{12}\,SMAX + \frac{1}{12}\,PSPC\right] \qquad \text{(B6)}$$

$$+ \frac{[BW + SMAX\,] * CWT}{BW}$$

For cable type (2):

$$q = \frac{1}{6} * PSPC * SNOW$$

$$+ \frac{[BW + SMAX] * [PSPC + SMAX] * CAMOWWT}{6 * BW} \qquad \text{(B7)}$$

$$+ \frac{[BW + SMAX] * CWT}{BW}$$

For cable type (3):

$$q = \left[\frac{PH}{\tan\,(ANGLE)} + \frac{1}{12}\,BW\right] * SNOW$$

$$+ \frac{[PSPC + SMAX] * CAMOWWT}{PSPC} * \left[\frac{PH}{\sin\,(ANGLE)} + \frac{13}{12}\,SMAX + \frac{1}{12}\,BW\right] \qquad \text{(B8)}$$

$$+ \frac{[PSPC + SMAX] * CWT}{PSPC}$$

For cable type (4):

$$q = \frac{[BW2 + BW1\,] * SNOW}{12}$$

$$+ \frac{[PSPC + SMAX] * CAMOWWT}{12 * PSPC} * [BW2 + BW1 + 2 * SMAX] \qquad \text{(B9)}$$

$$+ \frac{[PSPC + SMAX] * CWT}{PSPC}$$

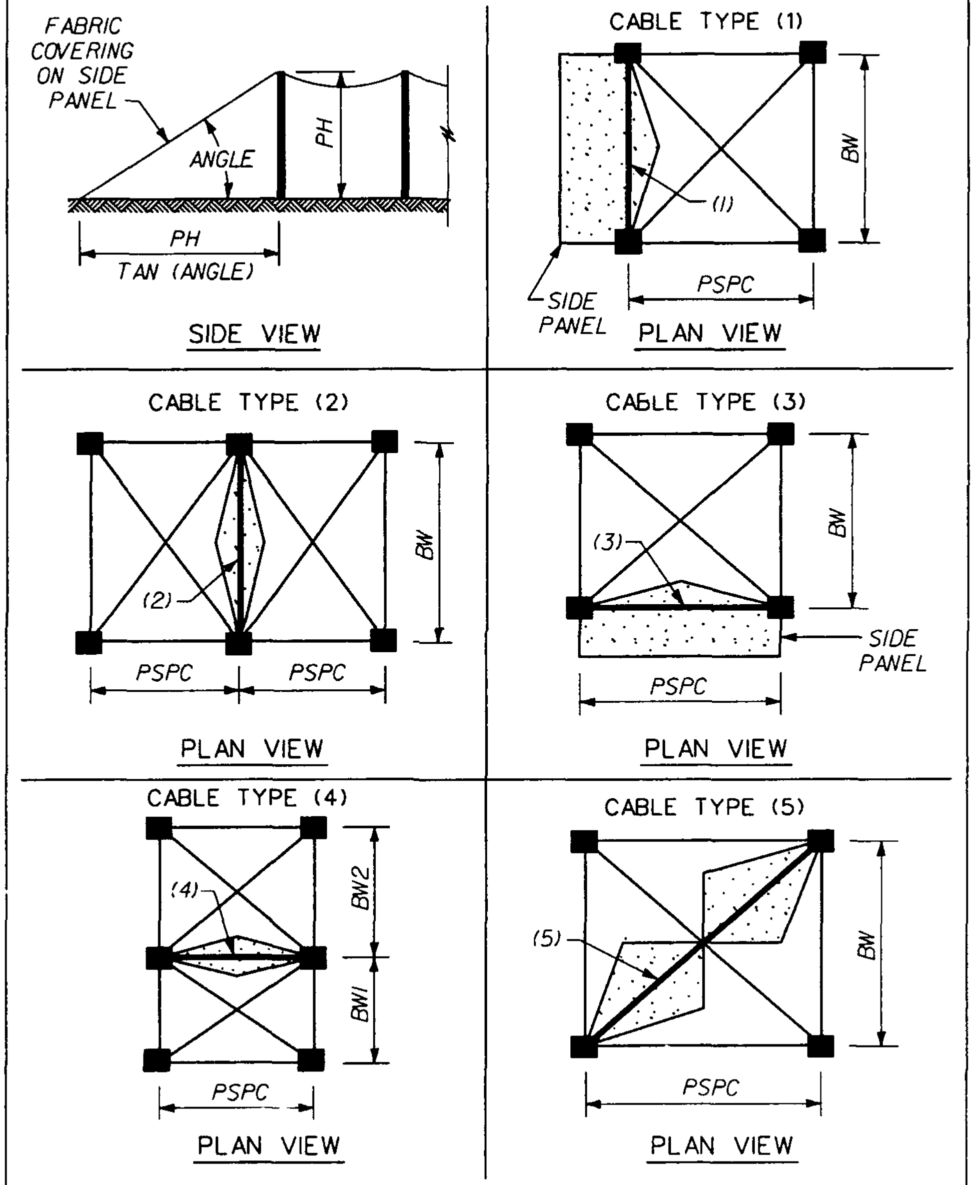

Figure B7. Cable types

For cable type (5):

$$q = \frac{BW * PSPC * SNOW + [BW + SMAX] * [PSPC + SMAX] * CAMOWWT}{3 * L} \tag{B10}$$

$$+ \frac{[L + SMAX] - CWT}{L}$$

WIND FORCES. Wind forces for this procedure are calculated by the mechanics of fluids equation given by Rouse and Howe (1953):

$$Wind\ Force\ (WF) = DC * \frac{1}{2} * 0.0024 * WV^2 * VA \tag{B11}$$

where WF is wind force in pounds, DC is the drag coefficient of the fabric, WV is the wind velocity in feet per second, and VA is the vertical projection area in square inches of the side panel. Once this force is known, it is divided by the length of the cable and added to the uniformly distributed load of the adjacent cable.

FINAL FORCES AND STRESSES COMPARED TO ALLOWABLES. Using the final deflections for a given trial pole spacing, equations 7-10 from Chapter 6 DESIGN Routine, and the following formulas taken from Leonard (1988), we can calculate the final tensile force in each cable:

$$\beta = \frac{q * (L_0 + \Delta L)}{2 * H} \tag{B12}$$

$$\gamma = \sinh^{-1}\left(\frac{\beta}{\sinh\beta} * \tan\theta\right) \tag{B13}$$

$$T = H_0 * \cosh\,[\gamma + \beta] \tag{B14}$$

The force "T" is the final tensile force along the length of the cable at the supports. This force can be compared to the allowable force in each cable which is the maximum force a cable can hold divided by the factor of safety.

The stress at the bottom of each support member is made up of three parts: (1) the vertical reaction for each node to the uniformly distributed load on each cable, which can be calculated as one half the length of the cable multiplied by the uniformly distributed load, (2) the moment caused by the deflections at the top of the pole, which are easily calculated for the cantilever, and (3) the vertical component of guy forces at exterior joints if guys are used, which is calculated by equation 2 in Chapter 6 DESIGN Routine.

References.

Leonard, W. M. (1988). *Tension structures.* McGraw-Hill Book Company, New York, NY.

Rouse, H. and Howe, J. W. (1953). *Basic mechanics of fluids.* John Wiley and Sons, New York, NY.

Appendix C
Program Source Codes

The source codes for the two FORTRAN routines developed for the Camouflage Program are included in Appendix C. The DESIGN routine (Chapter 6 DESIGN Routine) accomplishes the complete structural design for the project. The DRAW routine (Chapter 9 OUTPUT Routine) provides the plan view of the structure on the monitor screen and, optionally, on the printer.

The source codes for the DESIGN routine are provided below:

```
C TENSION STRUCTURE DESIGN (BRITT SIMMONS)       11 MAR 91
C REVISED  28 MAR 91
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FKO   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,HO    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,SO    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      DO 10 I=1,242
      DELT(I)=0.0
   10 CONTINUE
      DO 20 I=1,128
      UOLD(I)=0.0
   20 CONTINUE
      READ(5,*)NB
      DO 30 I=1,NB
      READ(5,*)BW(I)
   30 CONTINUE
      READ(5,*)TL
      READ(5,*)PH,PE,PI,PA,PFY,PW,PMSPC
      READ(5,*)CE,CA,TMAX,SMAX,CWT
      READ(5,*)ANGLE
      READ(5,*)CAMOWT,SNOW,WINDV,EMAX
      READ(5,32)GUY
   32 FORMAT(A1)
      READ(5,*)FS
      READ(5,*)DRAG
C        *****  CONVERT TO POUND AND SQUARE INCH UNITS
      PE=PE*10.0**6
      PFY=PFY*10.0**3
      CE=CE*10.0**6
      TMAX=TMAX*10.0**3
C        ***** CONVERSIONS,LBS & INCHES & RADIANS (FPS  FOR WIND
VELOCITY)
      DO 40 I=1,NB
      BW(I)=BW(I)*12.0
   40 CONTINUE
      TL=TL*12.0
      PH=PH*12.0
      PMSPC=PMSPC*12.0
      SMAX=SMAX*12.0
      CWT=CWT/12.0
      ANGLE=ANGLE*3.141592654/180.0
      CAMOWT=CAMOWT/144.0
      SNOW=SNOW/144.0
      WINDV=WINDV*5280.0/3600.0
      PR=SQRT(PI/PA)
      TMAX=TMAX/FS
      PFY=PFY/FS
      IF(FS.GT.2.0)THEN
          PFB=3.141592654**2*PE/FS/(2.0*PH/PR)**2
        ELSE
          PFB=3.141592654**2*PE/2.0/(2.0*PH/PR)**2
        ENDIF
      NSPCH=INT(TL/PMSPC)
      FNSPCL=0.9
      NSPC=INT(TL/PMSPC)
      WINDS=1.0
      TMMAX=0.0
      GTMAX=0.0
  200 PSPC=TL/NSPC
      NPX=NSPC+1
      NPY=NB+1
      NP=NPX*NPY
      KINC=16
      DO 210 I=1,2*NP
      U(I)=0.0
  210 CONTINUE
      CALL S1000
      IF(GUY.EQ.'Y')CALL S4000
  240 CALL S5000
```

```
      CALL S14000
      CALL S6000
      CALL S9000
      CALL S10000(EMAX)
      IF(GOOD.EQ.1.0)GO TO 290
      CALL S11000
      KINC=INT(KINC/2)
      IF(KINC.EQ.0)KINC=1
      GO TO 240
  290 CALL S13000
      IF(GOOD.EQ.1.0)GO TO 320
      IF(NSPC.EQ.INT(TL/PMSPC))THEN
          WRITE(6,*) 'SELECTED POLES WILL NOT SUPPORT STRUCTURE.'
          STOP
      ENDIF
      IF(NSPCH.EQ.(NSPC+1))THEN
          NSPC=NSPCH
          GO TO 365
      ENDIF
      FNSPCL=NSPC
      NSPC=NSPC+INT((NSPCH-INT(FNSPCL+0.2))/2.0+0.51)
      GO TO 200
  320 CALL S12000
      IF(GOOD.EQ.1.0)GO TO 350
      IF(NSPC.EQ.INT(TL/PMSPC))THEN
          WRITE(6,*) 'SELECTED CABLES FAIL AT MINIMUM POLE
SPACING.'
          STOP
      ENDIF
      IF(NSPCH.EQ.(NSPC+1))THEN
          NSPC=NSPCH
          GO TO 362
      ENDIF
      FNSPCL=NSPC
      NSPC=NSPC+INT((NSPCH-INT(FNSPCL+0.2))/2.0+0.51)
      GO TO 200
  350 IF(NSPC.EQ.(FNSPCL+1.0))GO TO 362
      NSPCH=NSPC
      NSPC=NSPC-INT((NSPCH-INT(FNSPCL+0.2))/2.0+0.51)
      GO TO 200
  362 CALL S19000
      IF(GUY.EQ.'Y')CALL S20000
  365 WINDS=2.0
  370 KINC=16
      NPX=NSPC+1
      NPY=NB+1
      NP=NPX*NPY
      DO 380 I=1,2*NP
      U(I)=0.0
  380 CONTINUE
```

```
      CALL S1000
      IF(GUY.EQ.'Y')CALL S4000
  400 CALL S5000
      CALL S15000
      CALL S6000
      CALL S9000
      CALL S10000(EMAX)
      IF(GOOD.EQ.1.0)GO TO 450
      CALL S11000
      KINC=INT(KINC/2)
      IF(KINC.EQ.0)KINC=1
      GO TO 400
  450 CALL S13000
      IF(GOOD.EQ.1.0)GO TO 475
      IF(NSPC.EQ.INT(TL/PMSPC))THEN
          WRITE(6,*) 'SELECTED POLES WILL NOT SUPPORT STRUCTURE.'
          STOP
        ENDIF
      NSPC=NSPC+1
      PSPC=TL/NSPC
      TMMAX=0.0
      GTMAX=0.0
      NREDO=1
      GO TO 370
  475 CALL S12000
      IF(GOOD.EQ.1.0)GO TO 497
      IF(NSPC.EQ.INT(TL/PMSPC))THEN
          WRITE(6,*) 'SELECTED CABLES FAIL AT MINIMUM POLE
SPACING.'
          STOP
        ENDIF
      NSPC=NSPC+1
      PSPC=TL/NSPC
      TMMAX=0.0
      GTMAX=0.0
      NREDO=1
      GO TO 370
  497 CALL S19000
      IF(GUY.EQ.'Y')CALL S20000
      WINDS=3.0
  505 KINC=16
      NPX=NSPC+1
      NPY=NB+1
      NP=NPX*NPY
      DO 510 I=1,2*NP
      U(I)=0.0
  510 CONTINUE
      CALL S1000
      IF(GUY.EQ.'Y')CALL S4000
  535 CALL S5000
```

```
      CALL S16000
      CALL S6000
      CALL S9000
      CALL S10000(EMAX)
      IF(GOOD.EQ.1.0)GO TO 585
      CALL S11000
      KINC=INT(KINC/2)
      IF(KINC.EQ.0)KINC=1
      GO TO 535
  585 CALL S13000
      IF(GOOD.EQ.1.0)GO TO 610
      IF(NSPC.EQ.INT(TL/PMSPC))THEN
          WRITE(6,*) 'SELECTED POLES WILL NOT SUPPORT STRUCTURE.'
          STOP
        ENDIF
      NSPC=NSPC+1
      PSPC=TL/NSPC
      TMMAX=0.0
      GTMAX=0.0
      NREDO=2
      GO TO 505
  610 CALL S12000
      IF(GOOD.EQ.1.0)GO TO 635
      IF(NSPC.EQ.INT(TL/PMSPC))THEN
          WRITE(6,*) 'SELECTED CABLES FAIL AT MINIMUM POLE
SPACING.'
          STOP
        ENDIF
      NSPC=NSPC+1
      PSPC=TL/NSPC
      TMMAX=0.0
      GTMAX=0.0
      NREDO=2
      GO TO 505
  635 CALL S19000
      IF(GUY.EQ.'Y')CALL S20000
      CALL S21000(EMAX)
      CALL S22000
      OPEN (UNIT=12,FILE='DEFLECTS.DAT')
      DO 650 I=1,NP*2
 650  WRITE(12,9999)U(I)
9999  FORMAT(' ',F10.6)
      CLOSE (UNIT=12)
      OPEN (UNIT=12,FILE='DESIGN.DAT')
      WRITE(12,*)GOOD
      CLOSE (UNIT=12)
  999 STOP
      END
      SUBROUTINE S1000
C ***** SET UP CABLE INFORMATION DATA FILE WITHOUT GUYS
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,H0    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,S0    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      QMAX=0.0
C          PERIMETER           CROSS            INTERIOR VERT
      NC=((NPX-1)+(NPY-1))*2+(NPX-1)*(NPY-1)*2+(NPX-2)*(NPY-1)
     >+(NPX-1)*(NPY-2)
C       INTERIOR HORZ
      NG=0
      IF(GUY.EQ.'Y')NG=2*NPX+2*NPY
      KN=0.0
      DO 1720 IB=1,NPY
      DO 1710 IA=1,NPX
      IF(IB.NE.1)GO TO 1350
      IF(IA.NE.1)GO TO 1170
C       ***** CABLES      JOINT CLASS 1
      DO 1160 IC=1,3
      IF(IC.EQ.1)THEN
          KN=KN+1
          FL=BW(1)
          JNT1=1
          JNT2=NPX+1
          FXFL=0.0
          FYFL=1.0
          DLDX=0.0
          DLDY=-1.0
          CC=1.0
        ENDIF
      IF(IC.EQ.2)THEN
          KN=KN+1
          FL=SQRT(BW(1)**2+PSPC**2)
          JNT1=1
          JNT2=NPX+2
          FXFL=PSPC/FL
          FYFL=BW(1)/FL
          CC=5.0
          DLDX=-(SQRT((PSPC+1.0)**2+BW(1)**2)-FL)
          DLDY=-(SQRT(PSPC**2+(BW(1)+1.0)**2)-FL)
        ENDIF
      IF(IC.EQ.3)THEN
          KN=KN+1
          FL=PSPC
          JNT1=1
          JNT2=2
          FXFL=1.0
          FYFL=0.0
          DLDX=-1.0
          DLDY=0.0
          CC=3.0
        ENDIF
      CALL S2000
      CALL S3000
 1160 CONTINUE
      GO TO 1710
 1170 IF(IA.EQ.NPX)GO TO 1290
C         ***** CABLES      JOINT CLASS 2
      DO 1280 IC=1,4
      IF(IC.EQ.1)THEN
          KN=KN+1
          FL=SQRT(BW(1)**2+PSPC**2)
          JNT1=IA
          JNT2=NPX+IA-1
          FXFL=-1*PSPC/FL
          FYFL=BW(1)/FL
          DLDX=SQRT((PSPC+1.0)**2+BW(1)**2)-FL
          DLDY=-(SQRT(PSPC**2+(BW(1)+1.0)**2)-FL)
          CC=5.0
        ENDIF
      IF(IC.EQ.2)THEN
          KN=KN+1
          FL=BW(1)
          JNT1=IA
          JNT2=NPX+IA
          FXFL=0.0
          FYFL=1.0
          DLDX=0.0
          DLDY=-1.0
          CC=2.0
        ENDIF
      IF(IC.EQ.3)THEN
          KN=KN+1
          FL=SQRT(BW(1)**2+PSPC**2)
          JNT1=IA
          JNT2=NPX+IA+1
          FXFL PSPC/FL
          FYFL=BW(1)/FL
          CC=5.0
          DLDX=-(SQRT((PSPC+1)**2+BW(1)**2)-FL)
          DLDY= (SQRT(PSPC**2+(BW(1)+1)**2)-FL)
        ENDIF
      IF(IC.EQ.4)THEN
          KN=KN+1
          FL=PSPC
          JNT1=IA
          JNT2=IA+1
          FXFL=1.0
          FYFL=0.0
          DLDX=-1.0
          DLDY=0.0
          CC=3.0
        ENDIF
      CALL S2000
      CALL S3000
 1280 CONTINUE
      GO TO 1710
C ***** CABLES      JOINT CLASS 3
 1290 DO 1340 IC=1,2
      IF(IC.EQ.1)THEN
          KN=KN+1
          FL=SQRT(BW(1)**2+PSPC**2)
          JNT1=NPX
          JNT2=2*NPX-1
          FXFL=-1.0*PSPC/FL
          FYFL=BW(1)/FL
          DLDX=SQRT((PSPC+1.0)**2+BW(1)**2)-FL
          DLDY=-(SQRT(PSPC**2+(BW(1)+1.0)**2)-FL)
          CC=5.0
        ENDIF
      IF(IC.EQ.2)THEN
          KN=KN+1
          FL=BW(1)
          JNT1=NPX
          JNT2=2*NPX
          FXFL=0.0
          FYFL=1.0
          DLDX=0.0
          DLDY=-1.0
          CC=1.0
        ENDIF
      CALL S2000
      CALL S3000
 1340 CONTINUE
      GO TO 1710
 1350 IF(IB.EQ.NPY)GO TO 1630
      IF(IA.NE.1)GO TO 1450
C ***** CABLES      JOINT CLASS 4
      DO 1440 IC=1,3
      IF(IC.EQ.1)THEN
          KN=KN+1
```

```
            FL=BW(IB)
            JNT1=(IB-1)*NPX+1
            JNT2=IB*NPX+1
            FXFL=0.0
            FYFL=1.0
            DLDX=0.0
            DLDY=-1.0
            CC=1.0
          ENDIF
        IF(IC.EQ.2)THEN
            KN=KN+1
            FL=SQRT(BW(IB)**2+PSPC**2)
            JNT1=(IB-1)*NPX+1
            JNT2=IB*NPX+2
            FXFL=PSPC/FL
            FYFL=BW(IB)/FL
            CC=5.0
            DLDX=-(SQRT((PSPC+1.0)**2+BW(IB)**2)-FL,
            DLDY=-(SQRT(PSPC**2+(BW(IB)+1.0)**2)-FL)
          ENDIF
        IF(IC.EQ.3)THEN
            KN=KN+1
            FL=PSPC
            JNT1=(IB-1)*NPX+1
            JNT2=(IB-1)*NPX+2
            FXFL=1.0
            FYFL=0.0
            DLDX=-1.0
            DLDY=0.0
            CC=4.0
          ENDIF
        CALL S2000
        CALL S3000
   1440 CONTINUE
        GO TO 1710
   1450 IF(IA.EQ.NPX)GO TO 1570
  C ***** CABLES      JOINT CLASS 5
        DO 1560 IC=1,4
        IF(IC.EQ.1)THEN
            KN=KN+1
            FL=SQRT(BW(IB)**2+PSPC**2)
            JNT1=(IB-1)*NPX+IA
            JNT2=IB*NPX+IA-1
            FXFL=-1.0*PSPC/FL
            FYFL=BW(IB)/FL
            DLDX=SQRT((PSPC+1.0)**2+BW(IB)**2)-FL
            DLDY=-(SQRT(PSPC**2+(BW(IB)+1.0)**2)-FL)
            CC=5.0
          ENDIF
        IF(IC.EQ.2)THEN
```

```
          KN=KN+1
          FL=BW(IB)
          JNT1=(IB-1)*NPX+IA
          JNT2=IB*NPX+IA
          FXFL=0.0
          FYFL=1.0
          DLDX=0.0
          DLDY=-1.0
          CC=2.0
        ENDIF
      IF(IC.EQ.3)THEN
          KN=KN+1
          FL=SQRT(BW(IB)**2+PSPC**2)
          JNT1=(IB-1)*NPX+IA
          JNT2=IB*NPX+IA+1
          FXFL=PSPC/FL
          FYFL=BW(IB)/FL
          CC=5.0
          DLDX=-(SQRT((PSPC+1.0)**2+BW(IB)**2)-FL)
          DLDY=-(SQRT(PSPC**2+(BW(IB)+1.0)**2)-FL)
        ENDIF
      IF(IC.EQ.4)THEN
          KN=KN+1
          FL=PSPC
          JNT1=(IB-1)*NPX+IA
          JNT2=(IB-1)*NPX+IA+1
          FXFL=1.0
          FYFL=0.0
          DLDX=-1.0
          DLDY=0.0
          CC=4.0
        ENDIF
      CALL S2000
      CALL S3000
 1560 CONTINUE
      GO TO 1710
C ***** CABLES      JOINT CLASS 6
 1570 DO 1620 IC=1,2
      IF(IC.EQ.1)THEN
          KN=KN+1
          FL=SQRT(BW(IB)**2+PSPC**2)
          JNT1=IB*NPX
          JNT2=(IB+1)*NPX-1
          FXFL=-1.0*PSPC/FL
          FYFL=BW(IB)/FL
          DLDX=SQRT((PSPC+1)**2+BW(IB)**2)-FL
          DLDY=-(SQRT(PSPC**2+(BW(IB)+1.0)**2)-FL)
          CC=5.0
        ENDIF
      IF(IC.EQ.2)THEN
```

```
          KN=KN+1
          FL=BW(IB)
          JNT1=IB*NPX
          JNT2=(IB+1)*NPX
          FXFL=0.0
          FYFL=1.0
          DLDX=0.0
          DLDY=-1.0
          CC=1.0
        ENDIF
      CALL S2000
      CALL S3000
 1620 CONTINUE
      GO TO 1710
 1630 IF(IA.NE.1)GO TO 1670
C ***** CABLES      JOINT CLASS 7
      KN=KN+1
      FL=PSPC
      JNT1=(NPY-1)*NPX+1
      JNT2=(NPY-1)*NPX+2
      FXFL=1.0
      FYFL=0.0
      DLDX=-1.0
      DLDY=0.0
      CC=3.0
      CALL S2000
      CALL S3000
      GO TO 1710
 1670 IF(IA.EQ.NPX)GO TO 1710
C       ***** CABLES      JOINT CLASS 8
      KN=KN+1
      FL=PSPC
      JNT1=(NPY-1)*NPX+IA
      JNT2=(NPY-1)*NPX+IA+1
      FXFL=1.0
      FYFL=0.0
      DLDX=-1.0
      DLDY=0.0
      CC=3.0
      CALL S2000
      CALL S3000
C       ***** NO HORIZONTAL CABLES      JOINT CLASS 9
 1710 CONTINUE
 1720 CONTINUE
      RETURN
      END
 2000 SUBROUTINE S2000
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,H0    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,S0    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
C ***** DEFINE FUNCTIONS      HYPERBOLIC SINE, HYPERBOLIC COSINE
C ***** AND INVERSE HYPERBOLIC SINE
      FNSINH(X)=(EXP(X)-EXP(-1*X))/2.0
      FNCOSH(X)=(EXP(X)+EXP(-1*X))/2.0
      FASINH(X)=ALOG(X+SQRT(X**2+1.0))
C ***** CALCULATE Q, S0, AND H0  FOR EACH TYPE CABLE
      QADD=0.0
      IF(CC.NE.1.0)GO TO 2060
C ***** Q  FOR BW-LENGTH CABLE ON EXTERIOR
      Q=(PH/TAN(ANGLE)+PSPC/12.0)*SNOW
      Q=Q+(BW(IB)+SMAX)*CAMOWT*(PH/SIN(ANGLE)
     >+13.0*SMAX/12.0+PSPC/12.0)/BW(IB)
      Q=Q+(BW(IB)+SMAX)*CWT/BW(IB)
      CALL S18000
      GO TO 2280
 2060 IF(CC.NE.2.0)GO TO 2100
C ***** Q  FOR BW-LENGTH CABLE ON INTERIOR
      Q=PSPC*SNOW/6.0
      Q=Q+(BW(IB)+SMAX)*(PSPC+SMAX)*CAMOWT/6.0/BW(IB)
     >+(BW(IB)+SMAX)*CWT/BW(IB)
      GO TO 2280
 2100 IF(CC.NE.3.0)GO TO 2180
      IF(IB.NE.1)GO TO 2160
C       ***** Q  FOR PSPC-LENGTH CABLE ON EXTERIOR    DO B=1
      Q=(PH/TAN(ANGLE)+BW(1)/12.0)*SNOW

Q=Q+(PSPC+SMAX)*CAMOWT*(PH/SIN(ANGLE)+13*SMAX/12+BW(1)/12)/PSPC
      Q=Q+(PSPC+SMAX)*CWT/PSPC
      CALL S17000
      GO TO 2280
C       ***** Q  FOR PSPC-LENGTH CABLE ON EXTERIOR   FOR B=NPY
 2160 Q=(PH/TAN(ANGLE)+BW(NPY-1)/12.0)*SNOW
      Q=Q+(PSPC+SMAX)*CAMOWT*(PH/SIN(ANGLE)
     >+13.0*SMAX/12.0+BW(NPY-1)/12.0)/PSPC
      Q=Q+(PSPC+SMAX)*CWT/PSPC
      CALL S17000
      GO TO 2280
 2180 IF(CC.NE.4.0)GO TO 2220
C       ***** Q  FOR PSPC-LENGTH CABLE ON INTERIOR
      Q=SNOW*(BW(IB)+BW(IB-1))/12.0
      Q=Q+(PSPC+SMAX)*CAMOWT*(BW(IB)+BW(IB-1)
     >+2.0*SMAX)/12.0/PSPC+(PSPC+SMAX)*CWT/PSPC
     GO TO 2280
 2220 IF(CC.NE.5.0)GO TO 2260
C       ***** Q  FOR DIAGONAL
      Q=(BW(IB)*PSPC*SNOW+(BW(IB)+SMAX)*(PSPC+SMAX)*CAMOWT)/3.0/FL
     >+(FL+SMAX)*CWT/FL
      GO TO 2280
 2260 CONTINUE
C ***** Q  FOR GUY
      Q=CWT
 2280 CONTINUE
C ***** CALCULATE H0, S0, FK01, AND FK02
      IF(CC.LE.5.0.AND.Q.GT.QMAX)QMAX=Q
      IF(CC.LE.5.0)THEN
          THETA=0.0
        ELSE
          THETA=0.785398163
        ENDIF
      IF(CC.LE.5)THEN
          F0=0.8*SMAX/FL
        ELSE
          F0=SMAX/FL*(CWT/QMAX)
        ENDIF
      IF(WINDS.EQ.1.0.AND.CC.EQ.3.0.AND.IB.EQ.1)Q=Q+QADD
      IF(WINDS.EQ.2.0.AND.CC.EQ.1.0.AND.IA.EQ.1)Q=Q+QADD
      IF(WINDS.EQ.3.0.AND.CC.EQ.3.0.AND.IB.EQ.NPY)Q=Q+QADD
      H0=Q*FL/8.0/F0
      BETA0=Q*FL/2.0/H0
      X=BETA0*TAN(THETA)/FNSINH(BETA0)
      GAMMA0=FASINH(X)
      X=GAMMA0+BETA0
      Y=GAMMA0-BETA0
      S01=H0*(FNSINH(X)-FNSINH(Y))/Q
      SINX=FNSINH(X)
      COSX=FNCOSH(X)
      SINY=FNSINH(Y)
      COSY=FNCOSH(Y)
      S02=(FL/2.0/CA/CE)*(H0+H0**2*(SINX*COSX-SINY*COSY)/Q/FL)
      S0=S01-S02
      FK01=TAN(THETA)**2+Q*FL*(1.0+TAN(THETA)**2
     >+16.0*F0**2/3.0)/8.0/CA/CE/F0
      FK02=S0/FL-1.0-TAN(THETA)**2/2.0
     >-8.0*F0**2+3.0*Q*FL/16.0/CA/CE/F0
      RETURN
      END
```

```
       SUBROUTINE S3000
       COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)     ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)    ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,H0    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,S0    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)     ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
       CHARACTER*1 GUY
C ***** WRITE S CABLE INFORMATION AND DATA (ZZ ARRAY)
       IF(WINDS.EQ.1.0.AND.CC.EQ.3.0.AND.IB.EQ.1)Q=Q-QADD
       IF(WINDS.EQ.2.0.AND.CC.EQ.1.0.AND.IA.EQ.1)Q=Q-QADD
       IF(WINDS.EQ.3.0.AND.CC.EQ.3.0.AND.IB.EQ.NPY)Q=Q-QADD
       ZZ(KN,1)=JNT1
       ZZ(KN,2)=JNT2
       ZZ(KN,3)=FXFL
       ZZ(KN,4)=FYFL
       ZZ(KN,5)=DLDX
       ZZ(KN,6)=DLDY
       ZZ(KN,7)=CC
       ZZ(KN,8)=Q
       ZZ(KN,9)=QADD
       ZZ(KN,10)=S0
       ZZ(KN,11)=FL
       ZZ(KN,12)=H0
       ZZ(KN,13)=FK01
       ZZ(KN,14)=FK02
       RETURN
       END
```

```
      SUBROUTINE S4000
C ***** SET UP CABLE INFORMATION  FOR GUYS
C ***** BOTTOM LINE GUYS
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,H0    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,S0    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      DO 4010 I=1,NPX
      KN=KN+1
      FL=PH
      JNT1=I
      JNT2=-1000
      FXFL=0.0
      FYFL=-1.0
      DLDX=0.0
      DLDY=1.0
      CC=6.0
      CALL S2000
      CALL S3000
 4010 CONTINUE
C ***** TOP LINE GUYS
      DO 4020 I=1,NPX
      KN=KN+1
      FL=PH
      JNT1=NP-NPX+I
      JNT2=-2000
      FXFL=0.0
      FYFL=1.0
      DLDX=0.0
      DLDY=-1.0
      CC=7.0
      CALL S2000
      CALL S3000
 4020 CONTINUE
C ***** LEFT LINE GUYS
      DO 4030I=1,NPY
      KN=KN+1
      FL=PH
      JNT1=(I-1)*NPX+1
      JNT2=-3000
      FXFL=-1.0
      FYFL=0.0
      DLDX=1.0
      DLDY=0.0
      CC=8.0
      CALL S2000
      CALL S3000
 4030 CONTINUE
C ***** RIGHT LINE GUYS
      DO 4040 I=1,NPY
      KN=KN+1
      FL=PH
      JNT1=I*NPX
      JNT2=-4000
      FXFL=1.0
      FYFL=0.0
      DLDX=-1.0
      DLDY=0.0
      CC=9.0
      CALL S2000
      CALL S3000
 4040 CONTINUE
      RETURN
      END
      SUBROUTINE S5000
C ***** FROM CABLE INFORMATION     CREATE FORCE MATRIX
      COMMON ANGLE ,BETA  ,BW(10),CA     ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)     ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)    ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL   ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,H0    ,IA     ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA    ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR     ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI     ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,S0    ,SMAX  ,SNOW   ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)     ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      DO 5010 J=1,2*NP
      F(J)=0.0
 5010 CONTINUE
      DO 5190 I=1,NC+NG
      DELTAL=DELT(I)
      CALL S24000(I)
      FK0=1.0+(FK01-S0/(FL+DELTAL))/FK02
      H=H0*(1.0+FK0*DELTAL/FL)
      F(2*JNT1-1)=F(2*JNT1-1)+FXFL*H
      IF(JNT2.LT.0)GO TO 5160
      F(2*JNT2-1)=F(2*JNT2-1)-FXFL*H
 5160 F(2*JNT1)=F(2*JNT1)+FYFL*H
      IF(JNT2.LT.0)GO TO 5190
      F(2*JNT2)=F(2*JNT2)-FYFL*H
 5190 CONTINUE
      DO 5200 I=1,2*NP
      IF(ABS(F(I)).LT.1.0)F(I)=0.0
 5200 CONTINUE
      RETURN
      END
      SUBROUTINE S6000
C ***** SET UP STIFFNESS MATRIX  FOR STRUCTURE
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,HO    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,SO    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      FKP=3.0*PE*PI/PH**3
      NBA=4*(NPX+2)-1
      NHB=INT(NBA/2)+1
      DO 6020 I=1,2*NP
      DO 6010 J=1,NBA
      FK(I,J)=0.0
 6010 CONTINUE
 6020 CONTINUE
      DO 6030 I=1,2*NP
      FK(I,NHB)=FKP
 6030 CONTINUE
      DO 6545 J=1,2*NP
      IF(J.EQ.INT(J/2)*2)GO TO 6305
C ***** X-DIRECTION FORCES
      IF(F(J).LE.0.0)GO TO 6130
C ***** POSITIVE Fx
      DO 6115 I=1,NC+NG
      CALL S24000(I)
      IF(JNT1.NE.(J+1)/2)GO TO 6095
      IF(JNT2.EQ.-3000)CALL S7000(I,J)
      IF(JNT2.EQ.JNT1+1)GO TO 6090
      IF(JNT2.EQ.JNT1+NPX-1)CALL S7000(I,J)
 6090 GO TO 6115
 6095 IF(JNT2.NE.(J+1)/2)GO TO 6115
      IF(JNT1.EQ.JNT2-NPX+1)GO TO 6115
      IF(JNT1.EQ.JNT2-1)CALL S7000(I,J)
      IF(JNT1.EQ.JNT2-NPX-1)CALL S7000(I,J)
 6115 CONTINUE
      GO TO 6545
 6130 IF(F(J).GE.0.0)GO TO 6215
C ***** NEGATIVE Fx
      DO 6200 I=1,NC+NG
      CALL S24000(I)
      IF(JNT1.NE.(J+1)/2)GO TO 6185
      IF(JNT2.EQ.-4000)CALL S7000(I,J)
      IF(JNT2.EQ.JNT1+NPX+1)CALL S7000(I,J)
      IF(JNT2.EQ.JNT1+NPX-1)GO TO 6180
      IF(JNT2.EQ.JNT1+1)CALL S7000(I,J)
 6180 GO TO 6200
 6185 IF(JNT2.NE.(J+1)/2)GO TO 6200
      IF(JNT1.EQ.JNT2-1)GO TO 6200
      IF(JNT1.EQ.JNT2-NPX+1)CALL S7000(I,J)
 6200 CONTINUE
      GO TO 6545
 6215 CONTINUE
C REM ***** Fx=0
      DO 6290 I=1,NC+NG
      CALL S24000(I)
      IF(JNT1.NE.(J+1)/2)GO TO 6260
      IF(JNT2.EQ.JNT1+1)GO TO 6255
      IF(JNT2.EQ.JNT1+NPX-1)CALL S7000(I,J)
      IF(JNT2.EQ.JNT1+NPX+1)CALL S7000(I,J)
 6255 IF(JNT2.EQ.JNT1+1)CALL S7000(I,J)
 6260 IF(JNT2.NE.(J+1)/2)GO TO 6290
      IF(JNT1.EQ.JNT2-NPX+1)GO TO 6285
      IF(JNT1.EQ.JNT2-1)CALL S7000(I,J)
      IF(JNT1.EQ.JNT2-NPX-1)CALL S7000(I,J)
      IF(JNT1.EQ.JNT2-1)GO TO 6290
 6285 IF(JNT1.EQ.JNT2-NPX+1)CALL S7000(I,J)
 6290 CONTINUE
      GO TO 6545
 6305 CONTINUE
C ***** Y-DIRECTION FORCES
      IF(F(J).LE.0.0)GO TO 6390
C ***** POSITIVE Fy
      DO 6375 I=1,NC+NG
      CALL S24000(I)
      IF(JNT1.NE.J/2)GO TO 6350
      IF(JNT2.EQ.-1000)CALL S8000(I,J)
      GO TO 6375
 6350 IF(JNT2.NE.J/2)GO TO 6375
      IF(JNT1.EQ.JNT2-NPX-1)CALL S8000(I,J)
      IF(JNT1.EQ.JNT2-NPX)CALL S8000(I,J)
      IF(JNT1.EQ.JNT2-1)GO TO 6375
      IF(JNT1.EQ.JNT2-NPX+1)CALL S8000(I,J)
 6375 CONTINUE
      GO TO 6545
 6390 IF(F(J).GE.0.0)GO TO 6460
C ***** NEGATIVE Fy
      DO 6445 I=1,NC+NG
      CALL S24000(I)
      IF(JNT1.NE.J/2)GO TO 6445
      IF(JNT2.EQ.-2000)CALL S8000(I,J)
      IF(JNT2.EQ.JNT1+1)GO TO 6445
      IF(JNT2.EQ.JNT1+NPX-1)CALL S8000(I,J)
      IF(JNT2.EQ.JNT1+NPX)CALL S8000(I,J)
      IF(JNT2.EQ.JNT1+NPX+1)CALL S8000(I,J)
 6445 CONTINUE
      GO TO 6545
 6460 CONTINUE
C ***** Fy=0
       DO 6535 I=1,NC+NG
      CALL S24000(I)
      IF(JNT1.NE.J/2)GO TO 6510
      IF(JNT2.EQ.JNT1+1)GO TO 6505
      IF(JNT2.EQ.JNT1+NPX-1)CALL S8000(I,J)
      IF(JNT2.EQ.JNT1+NPX)CALL S8000(I,J)
      IF(JNT2.EQ.JNT1+NPX+1)CALL S8000(I,J)
 6505 GO TO 6535
 6510 IF(JNT2.NE.J/2)GO TO 6535
      IF(JNT1.EQ.JNT2-NPX-1)CALL S8000(I,J)
      IF(JNT1.EQ.JNT2-NPX)CALL S8000(I,J)
      IF(JNT1.EQ.JNT2-1)GO TO 6535
      IF(JNT1.EQ.JNT2-NPX+1)CALL S8000(I,J)
 6535 CONTINUE
 6545 CONTINUE
      RETURN
      END
      SUBROUTINE S7000(I,J)
C ***** MODIFY 'K' MATRIX  FOR X-FORCES
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,H0    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,S0    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      DELTAL=DELT(I)
      FK0=1.0+(FK01-S0/(FL+DELTAL+KINC*ABS(DLDX)))/FK02
      FKCX=H0*(1.0+FK0*(DELTAL+KINC*ABS(DLDX))/FL)/KINC
      FKC3=0.0
      FKC4=0.0
      IF(JNT1.EQ.(J+1)/2)THEN
          FKC1=FKCX*ABS(FXFL)
          FKC2=-FKC1
        ENDIF
      IF(JNT2.LT.0)GO TO 7120
      IF(JNT1.EQ.(J+1)/2.AND.JNT2.EQ.JNT1+NPX-1)THEN
          FKC3=-FKCX*ABS(FYFL)
          FKC4=-FKC3
        ENDIF
      IF(JNT1.EQ.(J+1)/2.AND.JNT2.EQ.JNT1+NPX+1)THEN
          FKC3=FKCX*ABS(FYFL)
          FKC4=-FKC3
        ENDIF
      IF(JNT2.EQ.(J+1)/2)THEN
          FKC2=FKCX*ABS(FXFL)
          FKC1=-FKC2
        ENDIF
      IF(JNT2.EQ.(J+1)/2.AND.JNT1.EQ.JNT2-NPX+1)THEN
          FKC3=-FKCX*ABS(FYFL)
          FKC4=-FKC3
        ENDIF
      IF(JNT2.EQ.(J+1)/2.AND.JNT1.EQ.JNT2-NPX-1)THEN
          FKC3=FKCX*ABS(FYFL)
          FKC4=-FKC3
        ENDIF
 7120 JB=NHB+J-2*JNT1+1
      FK(2*JNT1-1,JB)=FK(2*JNT1-1,JB)+FKC1
      IF(JNT2.LT.0)GO TO 7170
      JB=NHB+J-2*JNT2+1
      FK(2*JNT2-1,JB)=FK(2*JNT2-1,JB)+FKC2
      JB=NHB+J-2*JNT1
      FK(2*JNT1,JB)=FK(2*JNT1,JB)+FKC3
      JB=NHB+J-2*JNT2
      FK(2*JNT2,JB)=FK(2*JNT2,JB)+FKC4
 7170 RETURN
      END
      SUBROUTINE S8000(I,J)
C ***** MODIFY 'K' MATRIX  FOR Y-FORCES
      COMMON ANGLE ,BETA  ,BW(10),CA     ,CAMOWT,CC     ,
     >CE     ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX   ,
     >DLDY   ,DRAG  ,F(128),FK(128,39)   ,FK0   ,FK01   ,
     >FK02   ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD   ,
     >GTMAX  ,GUY   ,H     ,H0    ,IA    ,IB    ,JNT1   ,
     >JNT2   ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG     ,
     >NHB    ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC   ,
     >PA     ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q      ,
     >QADD   ,QMAX  ,S0    ,SMAX  ,SNOW  ,THETA ,TL     ,
     >TMAX   ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV  ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      DELTAL=DELT(I)
      FK0=1.0+(FK01-S0/(FL+DELTAL+KINC*ABS(DLDY)))/FK02
      FKCY=H0*(1.0+FKC*(DELTAL+KINC*ABS(DLDY))/FL)/KINC
      FKC3=0.0
      FKC4=0.0
      IF(JNT1.EQ.J/2)THEN
          FKC1=FKCY*ABS(FYFL)
          FKC2=-FKC1
        ENDIF
      IF(JNT2.LT.0)GO TO 8120
      IF(JNT1.EQ.J/2.AND.JNT2.EQ.JNT1+NPX-1)THEN
          FKC3=-FKCY*ABS(FXFL)
          FKC4=-FKC3
        ENDIF
      IF(JNT1.EQ.J/2.AND.JNT2.EQ.JNT1+NPX+1)THEN
          FKC3=FKCY*ABS(FXFL)
          FKC4=-FKC3
        ENDIF
      IF(JNT2.EQ.J/2)THEN
          FKC2=FKCY*ABS(FYFL)
          FKC1=-FKC2
        ENDIF
      IF(JNT2.EQ.J/2.AND.JNT1.EQ.JNT2-NPX+1)THEN
          FKC3=-FKCY*ABS(FXFL)
          FKC4=-FKC3
        ENDIF
      IF(JNT2.EQ.J/2.AND.JNT1.EQ.JNT2-NPX-1)THEN
          FKC3=FKCY*ABS(FXFL)
          FKC4=-FKC3
        ENDIF
 8120 JB=NHB+J-2*JNT1
      FK(2*JNT1,JB)=FK(2*JNT1,JB)+FKC1
      IF(JNT2.LT.0)GO TO 8170
      JB=NHB+J-2*JNT2
      FK(2*JNT2,JB)=FK(2*JNT2,JB)+FKC2
      JB=NHB+J-2*JNT1+1
      FK(2*JNT1-1,JB)=FK(2*JNT1-1,JB)+FKC3
      JB=NHB+J-2*JNT2+1
      FK(2*JNT2-1,JB)=FK(2*JNT2-1,JB)+FKC4
 8170 RETURN
      END
      SUBROUTINE S9000
C ***** GAUSSIAN ELIMINATION
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FKO   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,HO    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,SO    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      NR=2*NP
      DO 9010 I=1,NR
      U(I)=F(I)
 9010 CONTINUE
      CALL SOLVE(NR,NBA,NHB,FK,U)
      RETURN
      END
      SUBROUTINE S10000(EMAX)
C ***** COMPARE NEW U WITH OLD U
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,H0    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,S0    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      GOOD=1.0
      DO 10010 I=1,2*NP
      DIFF=ABS(UOLD(I)-U(I))
      IF(ABS(UOLD(I)-U(I)).GT.EMAX)THEN
          GOOD=0.0
          GO TO 10100
        ENDIF
10010 CONTINUE
      GO TO 10180
10100 DO 10110 I=1,2*NP
      U(I)=(UOLD(I)+U(I))/2.0
      UOLD(I)=U(I)
10110 CONTINUE
10180 RETURN
      END
      SUBROUTINE S11000
C ***** CALCULATE DELTAL  FOR EACH CABLE
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FKO   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,HO    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,SO    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      DO 11010 I=1,NC+NG
      CALL S24000(I)
      NSUB1=JNT1*2
      NSUB2=JNT2*2
      IF(JNT2.LT.0)THEN
          DELTAL=U(NSUB1-1)*DLDX+U(NSUB1)*DLDY
      ELSE

DELTAL=(U(NSUB1-1)-U(NSUB2-1))*DLDX+(U(NSUB1)-U(NSUB2))*DLDY
      ENDIF
      DELT(I)=DELTAL
11010 CONTINUE
      RETURN
      END
      SUBROUTINE S12000
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,H0    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,S0    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
C ***** DEFINE FUNCTIONS      HYPERBOLIC SINE, HYPERBOLIC COSINE
C ***** AND INVERSE HYPERBOLIC SINE
      FNSINH(X)=(EXP(X)-EXP(-X))/2.0
      FNCOSH(X)=(EXP(X)+EXP(-X))/2.0
      FASINH(X)=ALOG(X+SQRT(X**2+1.0))
C ***** CHECK TENSIONS
      GOOD=1.0
      DO 12010 I=1,NC+NG
      CALL S24000(I)
      DELTAL=DELT(I)
      FK0=1.0+(FK01-S0/(FL+DELTAL))/FK02
      H=H0*(1.0+FK0*DELTAL/FL)
      BETA=Q*(FL+DELTAL)/2.0/H
      IF(CC.LE.5.0)THEN
         THETA=0.0
      ELSE
         THETA=0.785398163
      ENDIF
      X=TAN(THETA)*BETA/FNSINH(BETA)
      GAMMA=FASINH(X)
      X=GAMMA+BETA
      T1=H0*FNCOSH(X)
      X=GAMMA-BETA
      T2=H0*FNCOSH(X)
      IF(T1.GT.TMAX.OR.T2.GT.TMAX)THEN
         GOOD=0.0
         RETURN
      ENDIF
12010 CONTINUE
      RETURN
      END
      SUBROUTINE S13000
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FKO   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,HO    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,SO    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
C ***** DEFINE FUNCTIONS      HYPERBOLIC SINE, HYPERBOLIC COSINE
C ***** AND INVERSE HYPERBOLIC SINE
      FNSINH(X)=(EXP(X)-EXP(-1*X))/2.0
      FNCOSH(X)=(EXP(X)+EXP(-1*X))/2.0
      FASINH(X)=ALOG(X+SQRT(X**2+1.0))
C ***** CHECK POLE STRESSES AT BASE
      GOOD=1.0
      DO 13200 I=1,NP
      V=0.0
      NSUB=I*2
      FADD=FKP*SQRT(U(NSUB-1)**2+U(NSUB)**2)
      DO 13190 J=1,NC+NG
      CALL S24000(J)
      IF(JNT1.NE.J.AND.JNT2.NE.J)GO TO 13190
      IF(CC.LE.5.0)THEN
          V=V+0.5*Q*FL
          GO TO 13190
      ENDIF
      DELTAL=DELT(J)
      FKO=1.0+(FK01-SO/(FL+DELTAL))/FK02
      H=HO*(1.0+FKO*DELTAL/FL)
      BETA=Q*(FL+DELTAL)/2.0/H
      IF(CC.LE.5.0)THEN
          THETA=0.0
      ELSE
           THETA=.785398163
      ENDIF
      X=TAN(THETA)*BETA/FNSINH(BETA)
      GAMMA=FASINH(X)
      X=GAMMA+BETA
      V=V+HO*FNSINH(X)
13190 CONTINUE
     PSTRES=V/PA+FADD*PH*PW/2/PI+V*SQRT(U(2*I-1)**2+U(2*I)**2)*PW/2/PI
     IF(PSTRES.GT.PFB)THEN
          GOOD=0.0
          RETURN
      ENDIF
```

```
13200 CONTINUE
      RETURN
      END
      SUBROUTINE S14000
C ***** MODIFY 'F' MATRIX  FOR WIND FROM BOTTOM OF PLAN
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,H0    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,S0    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      WINDF=DRAG*0.5*0.0024*WINDV**2*PH*PSPC/144.0
      DO 14010 I=1,NSPC
      NFSB=2*(I+1)
      F(2*I)=F(2*I)+WINDF/4.0
      F(NFSB)=F(NFSB)+WINDF/4.0
14010 CONTINUE
      RETURN
      END
      SUBROUTINE S15000
C ***** MODIFY 'F' MATRIX  FOR WIND FROM LEFT OF PLAN
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,H0    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,S0    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      DO 15010 I=1,NB
      NFSB1=2*NPX*(I-1)+1
      NFSB2=2*NPX*I+1
      WINDF=DRAG*0.5*0.0024*WINDV**2*PH*BW(I)/144.0
      F(NFSB1)=F(NFSB1)+WINDF/4.0
      F(NFSB2)=F(NFSB2)+WINDF/4.0
15010 CONTINUE
      RETURN
      END
      SUBROUTINE S16000
C ***** MODIFY 'F' MATRIX  FOR WIND FROM TOP OF PLAN
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,H0    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,S0    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      WINDF=DRAG*0.5*0.0024*WINDV**2*PH*PSPC/144.0
      DO 16010 I=1,NSPC
      NFSB1=2*(NP-NPX)+2*I
      NFSB2=2*(NP-NPX)+2*(I+1)
      F(NFSB1)=F(NFSB1)-WINDF/4.0
      F(NFSB2)=F(NFSB2)-WINDF/4.0
16010 CONTINUE
      RETURN
      END
      SUBROUTINE S17000
C ***** Q ADD  FOR WIND FROM TOP OR BOTTOM OF PLAN
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,H0    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,S0    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      WINDF=DRAG*0.5*0.0024*WINDV**2*PSPC*PH/144.0
      QADD=WINDF/FL
      RETURN
      END
      SUBROUTINE S18000
C ***** Q ADD  FOR WIND FROM LEFT OF PLAN
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,H0    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,S0    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      WINDF=DRAG*0.5*0.0024*WINDV**2*BW(IB)*PH/144.0
      QADD=WINDF/FL
      RETURN
      END
      SUBROUTINE S19000
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,H0    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,S0    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
C ***** DEFINE FUNCTIONS      HYPERBOLIC SINE, HYPERBOLIC COSINE
C ***** AND INVERSE HYPERBOLIC SINE
      FNSINH(X)=(EXP(X)-EXP(-1*X))/2.0
      FNCOSH(X)=(EXP(X)+EXP(-1*X))/2.0
      FASINH(X)=ALOG(X+SQRT(X**2+1.0))
C ***** MAXIMUM  MOMENT
      DO 19200 I=1,NP
      V=0.0
      FADD=FKP*SQRT(U(2*I-1)**2+U(2*I)**2)
      DO 19190 J=1,NC+NG
      CALL S24000(I)
      IF(JNT1.NE.J.AND.JNT2.NE.J)GO TO 19190
      IF(CC.LE.5.0)THEN
          V=V+.5*Q*FL
          GO TO 19190
        ENDIF
      DELTAL=DELT(J)
      FK0=1.0+(FK01-S0/(FL+DELTAL))/FK02
      H=H0*(1+FK0*DELTAL/FL)
      BETA=Q*(FL+DELTAL)/2.0/H
      IF(CC.LE.5.0)THEN
          THETA=0.0
        ELSE
          THETA=.785398163
        ENDIF
      X=TAN(THETA)*BETA/FNSINH(BETA)
      GAMMA=FASINH(X)
      X=GAMMA+BETA
      V=V+H0*FNSINH(X)
19190 CONTINUE
      TMOM=V*SQRT(U(2*I-1)**2+U(2*I)**2)+FADD*PH
      IF(TMOM.GT.TMMAX)TMMAX=TMOM
19200 CONTINUE
      RETURN
      END
      SUBROUTINE S20000
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,H0    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,S0    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
C ***** DEFINE FUNCTIONS      HYPERBOLIC SINE, HYPERBOLIC COSINE
C ***** AND INVERSE HYPERBOLIC SINE
      FNSINH(X)=(EXP(X)-EXP(-1*X))/2.0
      FNCOSH(X)=(EXP(X)+EXP(-1*X))/2.0
      FASINH(X)=ALOG(X+SQRT(X**2+1.0))
C ***** MAXIMUM GUY TENSIONS
      DO 20010 I=1,NC
      CALL S24000(I)
20010 CONTINUE
      DO 20020 I=NC+1,NC+NG
      CALL S24000(I)
      DELTAL=DELT(I)
      FK0=1.0+(FK01-S0/(FL+DELTAL))/FK02
      H=H0*(1.0+FK0*DELTAL/FL)
      BETA=Q*(FL+DELTAL)/2.0/H
      THETA=.785398163
      X=TAN(THETA)*BETA/FNSINH(BETA)
      GAMMA=FASINH(X)
      X=GAMMA+BETA
      GT1=H0*FNCOSH(X)
      X=GAMMA-BETA
      GT2=H0*FNCOSH(X)
      IF(GT1.GT.GTMAX)GTMAX=GT1
      IF(GT2.GT.GTMAX)GTMAX=GT2
20020 CONTINUE
      END
      SUBROUTINE S21000(EMAX)
C ***** REDO
      COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
     >CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)   ,FK0   ,FK01  ,
     >FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
     >GTMAX ,GUY   ,H     ,H0    ,IA    ,IB    ,JNT1  ,
     >JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
     >NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
     >PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
     >QADD  ,QMAX  ,S0    ,SMAX  ,SNOW  ,THETA ,TL    ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      DO 21200 KREDON=1,NREDO
      WINDS=KREDON
      KINC=16
      NPX=NSPC+1
      NPY=NB+1
      NP=NPX*NPY
      DO 21080 I=1,2*NP
      U(I)=0
21080 CONTINUE
      CALL S1000
      IF(GUY.EQ.'Y')CALL S4000
21090 CALL S5000
      IF(WINDS.EQ.1.0)THEN
          CALL S14000
        ELSE
          CALL S15000
        ENDIF
      CALL S6000
      CALL S9000
      CALL S10000(EMAX)
      IF(GOOD.EQ.1.0)GO TO 21190
      CALL S11000
      KINC=INT(KINC/2)
      IF(KINC.EQ.0)KINC=1
      GO TO 21090
21190 CALL S19000
      IF(GUY.EQ.'Y')CALL S20000
21200 CONTINUE
      RETURN
      END
      SUBROUTINE S22000
C ***** CREATE FINAL OUTPUT FILE (DESOUTPT.DAT)
      COMMON ANGLE ,BETA  ,BW(10),CA     ,CAMOWT,CC     ,
     >CE    ,CN    ,CWT   ,DELT(242)     ,DELTAL,DLDX   ,
     >DLDY  ,DRAG  ,F(128),FK(128,39)    ,FK0   ,FK01   ,
     >FK02  ,FKP   ,FL    ,FS     ,FXFL  ,FYFL  ,GAMMA ,GOOD   ,
     >GTMAX ,GUY   ,H     ,H0     ,IA    ,IB    ,JNT1   ,
     >JNT2  ,KINC  ,KN    ,NB     ,NBA   ,NC    ,NG     ,
     >NHB   ,NP    ,NPX   ,NPY    ,NR    ,NREDO ,NSPC   ,
     >PA    ,PE    ,PFB   ,PH     ,PI    ,PSPC  ,PW    ,Q      ,
     >QADD  ,QMAX  ,S0    ,SMAX   ,SNOW  ,THETA ,TL     ,
     >TMAX  ,TMMAX ,U(128),UOLD(128)     ,WINDF ,WINDS ,WINDV ,
     >ZZ(242,14)
      CHARACTER*1 GUY
      OPEN (UNIT=16,FILE='DESOUTPT.DAT')
      WRITE(16,22999)NSPC
22999 FORMAT(I5)
      WRITE(16,22998)PSPC
22998 FORMAT(F15.4,2I10,F10.2)
      WRITE(16,22998)TMMAX
      WRITE(16,22998)GTMAX
C     DO 22140 I=1,NPY
C     DO 22130 J=1,NC
C     CALL S24000(J)
C     IF(JNT1.EQ.(I-1)*NPX+1.AND.CC.EQ.3.0)THEN
C         WRITE(16,22998)S0,JNT1,JNT2,CC
C         GO TO 22140
C       ENDIF
C     IF(JNT1.EQ.(I-1)*NPX+1.AND.CC.EQ.4.0)THEN
C         WRITE(16,22998)S0,JNT1,JNT2,CC
C         GO TO 22140
C       ENDIF
22130 CONTINUE
22140 CONTINUE
C     DO 22230 I=1,NB
C     DO 22220 J=1,NC
C     CALL S24000(J)
C     IF(JNT1.EQ.(I-1)*NPX+1.AND.CC.EQ.1.0)WRITE(16,22998)S0,JNT1,
C    1    JNT2,CC
C     IF(JNT1.EQ.(I-1)*NPX+2.AND.CC.EQ.2)THEN
C         WRITE(16,22998)S0,JNT1,JNT2,CC
C         GO TO 22230
C       ENDIF
22220 CONTINUE
22230 CONTINUE
      DO 22240 I=1,NC
      CALL S24000(I)
      WRITE(16,22998)S0,JNT1,JNT2,CC
22240 CONTINUE
      CALL S24000(I)
```

```
WRITE(16,22998)S0,JNT1,JNT2,CC
CLOSE (UNIT=16)
RETURN
END
```

```
 SUBROUTINE S24000(I)
 COMMON ANGLE ,BETA  ,BW(10),CA    ,CAMOWT,CC    ,
>CE    ,CN    ,CWT   ,DELT(242)    ,DELTAL,DLDX  ,
>DLDY  ,DRAG  ,F(128),FK(128,39)   ,FKO   ,FK01  ,
>FK02  ,FKP   ,FL    ,FS    ,FXFL  ,FYFL  ,GAMMA ,GOOD  ,
>GTMAX ,GUY   ,H     ,HO    ,IA    ,IB    ,JNT1  ,
>JNT2  ,KINC  ,KN    ,NB    ,NBA   ,NC    ,NG    ,
>NHB   ,NP    ,NPX   ,NPY   ,NR    ,NREDO ,NSPC  ,
>PA    ,PE    ,PFB   ,PH    ,PI    ,PSPC  ,PW    ,Q     ,
>QADD  ,QMAX  ,SO    ,SMAX  ,SNOW  ,THETA ,TL    ,
>TMAX  ,TMMAX ,U(128),UOLD(128)    ,WINDF ,WINDS ,WINDV ,
>ZZ(242,14)
 CHARACTER*1 GUY
 JNT1=ZZ(I,1)
 JNT2=ZZ(I,2)
 FXFL=ZZ(I,3)
 FYFL=ZZ(I,4)
 DLDX=ZZ(I,5)
 DLDY=ZZ(I,6)
 CC=ZZ(I,7)
 Q=ZZ(I,8)
 QADD=ZZ(I,9)
 SO=ZZ(I,10)
 FL=ZZ(I,11)
 HO=ZZ(I,12)
 FK01=ZZ(I,13)
 FK02=ZZ(I,14)
 RETURN
 END
```

```
      SUBROUTINE SOLVE(NR,NB,NHB,A,B)
      DIMENSION A(128,1),B(1)
C  This subroutine will efficiently solve simultaneous equations with
C  banded coefficients.  Such a system of equations is illustrated below.
C
C    A(1,1)    A(1,2) A(1.3) 0      0      0        0       0
C    A(2,1)    A(2,2) A(2,3)     A(2,4)    0      0       0      0
C    A(3,1)    A(3,2) A(3,3)     A(3,4)  A(3,5)   0       0      0
C       0       A(4,2)       A(4,3)     A(4,4)  A(4,5)  A(4,6)   0      0
C       0          0         A(5,3)     A(5,4)  A(5,5)  A(5,6) A(5,7)      0
C       0          0           0        A(6,4)  A(6,5)  A(6,6) A(6,7) A(6,8)
C       0          0           0           0    A(7,5)  A(7,6) A(7,7) A(7,8)
C       0          0           0           0       0     A(8,6) A(8,7) A(8,8)
C
C  The above coefficients would be stored in an (8,5) array as follows:
C
C        0               0          A(1,1) A(1,2) A(1,3)
C        0            A(2,1) A(2,2) A(2,3) A(2,4)
C    A(3,1)    A(3,2) A(3,3) A(3,4) A(3,5)
C    A(4,2)    A(4,3) A(4,4) A(4,5) A(4,6)
C    A(5,3)    A(5,4) A(5,5) A(5,6) A(5,7)
C    A(6,4)    A(6,5) A(6,6) A(6,7) A(6,8)
C    A(7,5)    A(7,6) A(7,7) A(7,8)     0
C    A(8,6)    A(8,7) A(8,8)    0              0
C
C  NB - B.AND. WIDTH
C  NR - NUMBER OF EQUATIONS
C  NRA- ROW DIMENSION OF THE 'A' MATRIX
C
C  The band width should be the largest value calculated on any Ith row of
C  the following.
C                    2*(I-Jmin)+1   or   2*(Jmax-I)+1
C
C  The band width should always be an odd number.
C
C  The (I,I) coefficient should be stored in the banded array at
C  location (I,NHB) where NHB is INT(NB/2)+1
C  The coefficients of the equations array, A, should be dimensioned
C  A(NR,NB).  The right hand side of the equations array, B, should be
C  dimensioned B(NR).  The solution will be left in the array B.  The
C  array A and the original array B will be destroyed.
C
      DO 40550 KP=1,NR
      DO 40540 I=1,NHB
      IP=KP+I-1
      JP=NHB-I+1
      IF(IP.GT.NR)GO TO 40540
      DIV=A(IP,JP)
      IF(DIV.EQ.0.0)GO TO 40540
      DO 40510 J=1,NB
```

```
     A(IP,J)=A(IP,J)/DIV
40510 CONTINUE
      B(IP)=B(IP)/DIV
      IF(IP.EQ.KP)GO TO 40540
      DO 40520 J=1,NB
      IF(J-I+1.LT.1)GO TO 40520
      A(IP,J-I+1)=A(IP,J-I+1)-A(KP,J)
40520 CONTINUE
      B(IP)=B(IP)-B(KP)
40540 CONTINUE
40550 CONTINUE
      DO 40620 IB=1,NR
      I=NR-IB+1
      DO 40610 JB=2,NHB
      IF(I+JB-1.GT.NR)GO TO 40620
      B(I)=B(I)-A(I,NHB+JB-1)*B(I+JB-1)
40610 CONTINUE
40620 CONTINUE
      RETURN
      END
```

The source codes for the DRAW routine are provided below:

```
      CHARACTER*80 NAME
      CHARACTER*1 GUY,F
      DIMENSION Y(20),X(20),SCALES(16)
      DATA NOSCL,SCALES,SDIV/16,16.0,15.0,14.0,13.0,12.0,11.0,10.0,9.0,
     >8.0,7.0,6.0,5.0,4.0,3.0,2.0,1.0,128.0/
      READ(5,1)IPORT,MODEL,XSIZE,YSIZE
      CALL PLOTS(0,IPORT,MODEL)
      READ(5,4)NAME
    1 FORMAT(2I5,2F10.1)
    4 FORMAT(A80)
      OPEN (UNIT=8,FILE='DESINPUT.DAT',STATUS='OLD')
      Y(1)=0.0
      READ(8,*)NBAYS
      READ(8,*)(Y(I+1),I=1,NBAYS)
      DO 5 I=1,NBAYS
    5 Y(I+1)=Y(I+1)+Y(I)
      READ(8,*)WDTH
      READ(8,*)HIGH
      READ(8,11)F
      READ(8,11)F
      READ(8,11)F
   11 FORMAT(A1)
      READ(8,11)GUY
      IND=0
      IF(GUY.EQ.'Y')IND=1
      CLOSE (UNIT=8)
      OPEN (UNIT=8,FILE='DESOUTPT.DAT',STATUS='OLD')
      READ(8,*)NSPACE
      READ(8,*)SPACE
      CLOSE (UNIT=8)
      SPACE=SPACE/12.0
      X(1)=0.0
      DO 10 I=1,NSPACE
      X(I+1)=X(I)+SPACE
   10 CONTINUE
      HGTH=Y(NBAYS+1)+2.0*IND*HIGH
      WDTH=X(NSPACE+1)+2.0*IND*HIGH
      CALL BORDER(YSIZE,XSIZE)
      DO 20 I=1,NOSCL
      IF(HGTH*SCALES(I)/SDIV+2.25.LT.YSIZE.AND.WDTH*SCALES(I)/SDIV+2.0
     >.LT.XSIZE)GO TO 30
   20 CONTINUE
   30 SFFAC=SCALES(I)/SDIV
      XO=(XSIZE-WDTH*SFFAC)/2.0+0.5
      YO=(YSIZE-1.25-HGTH*SFFAC)/2.0+1.25
      XE=XO+WDTH*SFFAC
      DO 40 I=1,NBAYS+1
      YW=YO+IND*HIGH*SFFAC+Y(I)*SFFAC
      CALL PLOT(XO,YW,3)
      CALL PLOT(XE,YW,2)
   40 CONTINUE
      YE=YO+HGTH*SFFAC
      DO 50 I=1,NSPACE+1
      XW=XO+IND*HIGH*SFFAC+X(I)*SFFAC
      CALL PLOT(XW,YO,3)
      CALL PLOT(XW,YE,2)
   50 CONTINUE
      DO 60 I=1,NSPACE
      DO 60 J=1,NBAYS
      XL=XO+IND*HIGH*SFFAC+X(I)*SFFAC
      YB=YO+IND*HIGH*SFFAC+Y(J)*SFFAC
      XR=XO+IND*HIGH*SFFAC+X(I+1)*SFFAC
      YT=YO+IND*HIGH*SFFAC+Y(J+1)*SFFAC
      CALL PLOT(XL,YB,3)
      CALL PLOT(XR,YT,2)
      CALL PLOT(XL,YT,3)
      CALL PLOT(XR,YB,2)
      CALL CIRCLE(XL,YB,0.5,SFFAC,1)
      IF(J.EQ.NBAYS)CALL CIRCLE(XL,YT,0.5,SFFAC,1)
      IF(I.EQ.NSPACE)CALL CIRCLE(XR,YB,0.5,SFFAC,1)
      IF(I.EQ.NSPACE.AND.J.EQ.NBAYS)CALL CIRCLE(XR,YT,0.5,SFFAC,1)
      IF(I.EQ.1)THEN
          XD=XO-0.25
          D=Y(J+1)-Y(J)
          CALL DIMLIN(XD,YB,XD,YT,1,0,1,1,-1,0.25,1.0,0.1,0.1,D,0.1)
        ENDIF
      IF(J.EQ.1)THEN
          YD=YO-0.25
          D=X(I+1)-X(I)
          CALL DIMLIN(XL,YD,XR,YD,1,0,-1,1,-1,0.25,1.0,0.1,0.1,D,0.1)
        ENDIF
   60 CONTINUE
      CALL SYMBOL(0.5,0.5,0.1,NAME,0.0,80)
      CALL PLOT(0.0,0.0,999)
      STOP
      END
```

```
SUBROUTINE BORDER(YSIZE,XSIZE)
XLEFT=0.25
XRGHT=XSIZE-XLEFT
YBOT=0.25
YTOP=YSIZE-YBOT
CALL PLOT(XLEFT,YBOT,3)
CALL PLOT(XLEFT,YTOP,2)
CALL PLOT(XRGHT,YTOP,2)
CALL PLOT(XRGHT,YBOT,2)
CALL PLOT(XLEFT,YBOT,2)
RETURN
END
```

```
      SUBROUTINE CIRCLE(XC,YC,RAD,SF,IGILL)
      COMMON/PORT/IPORT,MODEL,SCL
      CHARACTER*32 ALPHA
      CHARACTER*80 OUTSTR
      DIMENSION X(36),Y(36)
      RADIUS=RAD*SF
      ARCINC=3.1415926/18.0
      DO 18 I=1,36
      ARC=ARCINC*I
      X(I)=XC+RADIUS*COS(ARC)
      Y(I)=YC+RADIUS*SIN(ARC)
   18 CONTINUE
      IF(IGILL.EQ.0)THEN
          CALL PLOT(X(36),Y(36),3)
          DO 20 I=1,36
          CALL PLOT(X(I),Y(I),2)
   20     CONTINUE
        ELSE
          CALL FILL(X,Y,36)
        ENDIF
      RETURN
      END
      SUBROUTINE DIMLIN(X1,Y1,X2,Y2,LNTYP,IHD,IOL,ILA,IRA,
     >SPACE,SF,HGTHR,XSTEPR,DIM,TOP)
      CHARACTER*13 STR
      DIMENSION X(17),Y(17)
C X1,Y1          LEFT END POINT  (PLOT COORDINATES)
C X2,Y2          RIGHT END POINT (PLOT COORDINATES)
C
C                              6-0
C LNTYP             |<------------->|   =1
C                 3-0
C                ---->|          |<--          =2
C                                         3-0
C                ---->|          |<---         =3
C
C IHD            0=NO HEADERS    1=HEADERS        .
C
C IOL            16-0                   =+1
C                |<------------->|
C                |               |
C                |               |
C                :               :
C                |               |
C                |       16-0    |
C                |<------------->|      =-1
C
C
C ARROW HEADS   > = -1        < = +1     0 = NO ARROW HEAD
C ILA            LEFT END ARROW HEAD
C IRA            RIGHT END ARRCW HEAD
C
C SPACE          DISTANCE FROM DRAWING LINE TO DIMENSION LINE
C SF             TEXT SCALE FACTOR
C HGTH           CHARACTER HEIGHT
C XSTEP          CHARACTER SPACE WIDTH
C DIM            ANNOTATION LENGTH (REAL WORLD)
C
```

```
C POINT LOCATIONS
C
C
C                     6                            15
C             8___    | 5    9____                 | 14 17____
C                 7-3<------------------------12<--16
C                     | 4                          | 13
C                     |                            |
C                     |                            |
C                     |                            |
C                     2                            11
C
C                     1                            10
C
C THE SYMBOL IS CONSTRUCTED HORIZONTALLY AND ROTATED ABOUT
C POINT 1. UNNEEDED PARTS ARE OMITTED.
C
      HGTH=HGTHR*SF
      RAD=HGTHR/4.0
      XSTEP=XSTEPR
      CALL FOOTS(DIM,STR,NC)
      DIST=SQRT((Y2-Y1)**2+(X2-X1)**2)
      IF(DIST.LE.0.0)RETURN
      ALPHA=ATAN2(Y2-Y1,X2-X1)
      DEGR=ALPHA*180.0/3.1415926
      X(1)=0.0
      Y(1)=0.0
      X(2)=X(1)
      Y(2)=Y(1)+SF*IOL
      X(6)=X(1)
      Y(6)=Y(1)+(SPACE+TOP)*SF*IOL
      X(3)=X(6)
      Y(3)=Y(6)-SF*IOL*TOP
      X(4)=X(3)+SF*ILA
      Y(4)=Y(3)-0.34*SF
      X(5)=X(3)+SF*ILA
      Y(5)=Y(3)+0.34*SF
      X(7)=X(3)-4.0*SF
      Y(7)=Y(3)
      X(8)=X(3)-(NC*XSTEP+1.5)*SF
      Y(8)=Y(3)+0.3*SF
      IF(LNTYP.EQ.2)X(7)=X(8)
      X(9)=DIST/2.0-NC/2.0*XSTEP*SF
      Y(9)=Y(3)+HGTHR*SF
      X(10)=X(1)+DIST
      Y(10)=0.0
      X(11)=X(10)
      Y(11)=Y(10)+SF*IOL
      X(15)=X(10)
      Y(15)=Y(10)+(SPACE+TOP)*SF*IOL
      X(12)=X(15)
      Y(12)=Y(15)-SF*IOL*TOP
      X(13)=X(12)+SF*IRA
      Y(13)=Y(12)-0.34*SF
      X(14)=X(12)+SF*IRA
      Y(14)=Y(12)+0.34*SF
      X(17)=X(12)+1.5*SF
      Y(17)=Y(12)+0.3*SF
      X(16)=X(12)+4.0*SF
      Y(16)=Y(12)
      IF(LNTYP.EQ.3)X(16)=X(17)+NC*XSTEP*SF
      DO 10 I=2,17
      BETA=ATAN2(Y(I),X(I))
      DIST=SQRT(Y(I)**2+X(I)**2)
      ANGL=ALPHA+BETA
      X(I)=X1+DIST*COS(ANGL)
      Y(I)=Y1+DIST*SIN(ANGL)
   10 CONTINUE
      X(1)=X1
      Y(1)=Y1
      IF(IHD.NE.0)THEN
          CALL PLOT(X(2),Y(2),3)
          CALL PLOT(X(6),Y(6),2)
        ENDIF
      IF(ILA.NE.0)THEN
C         CALL FILL(X(3),Y(3),3)
      CALL CIRCLE(X(3),Y(3),RAD,SF,0)
        ENDIF
      IF(LNTYP.GE.2)THEN
          CALL PLOT(X(7),Y(7),3)
          CALL PLOT(X(3),Y(3),2)
          IF(LNTYP.EQ.2)THEN
              CALL SYMBOL(X(8),Y(8),HGTH,STR,DEGR,NC)
            ENDIF
        ENDIF
      IF(LNTYP.EQ.1)THEN
          CALL SYMBOL(X(9),Y(9),HGTH,STR,DEGR,NC)
          CALL PLOT(X(3),Y(3),3)
          CALL PLOT(X(12),Y(12),2)
        ENDIF
      IF(IHD.NE.0)THEN
          CALL PLOT(X(11),Y(11),3)
          CALL PLOT(X(15),Y(15),2)
        ENDIF
      IF(IRA.NE.0)THEN
C         CALL FILL(X(12),Y(12),3)
          CALL CIRCLE(X(12),Y(12),RAD,SF,0)
        ENDIF
      IF(LNTYP.GE.2)THEN
          CALL PLOT(X(12),Y(12),3)
```

```
    CALL PLOT(X(16),Y(16),2)
    IF(LNTYP.EQ.3)THEN
        CALL SYMBOL(X(17),Y(17),HGTH,STR,DEGR,NC)
      ENDIF
  ENDIF
RETURN
END
```

```
      SUBROUTINE FOOTS(DIM,STR,NC)
      DIMENSION NFC(15)
      CHARACTER*1 NUM(10)
      CHARACTER*13 STR,BLK
      CHARACTER*5 FRACT(15)
      DATA BLK/'             '/
      DATA NUM/'0','1','2','3','4','5','6','7','8','9'/
      DATA FRACT/'1/16','1/8','3/16','1/4','5/16','3/8','7/16',
     >'1/2','9/16','5/8','11/16','3/4','13/16','7/8','15/16'/
      DATA NFC/4,3,4,3,4,3,4,3,6,3,5,3,5,3,5/
      DIST=DIM+0.002604167
      STR=BLK
      NC=0
      DIV=1000.0
      DO 100 I=1,4
      IF(DIST.LT.DIV.AND.NC.EQ.0)GO TO 90
      NT=DIST/DIV
      NC=NC+1
      STR(NC:NC)=NUM(NT+1)
      DIST=DIST-NT*DIV
   90 DIV=DIV/10.0
  100 CONTINUE
      IF(NC.GT.0)THEN
          NC=NC+1
          STR(NC:NC)='-'
        ENDIF
      DIST=DIST*12.0
      DIV=10.0
      DO 200 I=1,2
      IF(DIST.LT.DIV.AND.I.EQ.1)GO TO 190
      NT=DIST/DIV
      NC=NC+1
      STR(NC:NC)=NUM(NT+1)
      DIST=DIST-NT*DIV
  190 DIV=DIV/10.0
  200 CONTINUE
      N16=DIST/0.0625
      IF(NC.GT.0.AND.N16.GT.0)THEN
          NC=NC+1
          STR(NC:NC)=' '
        ENDIF
      IF(N16.GT.0)THEN
          NS=NC+1
          NC=NC+NFC(N16)
          STR(NS:NC)=FRACT(N16)
        ENDIF
      RETURN
      END
```

Appendix D Sample Program Output

```
              FIXED FACILITY SUPPORT SYSTEMS     PROJECT # TRIAL
                    Site Specific Data          REVISION # 0

     SOILS INVESTIGATION and FOUNDATION DATA

              cohesion (psf):  400.
              unit weight (pcf):  120.
              angle of internal friction (degrees):  25.0
              diameter of boring for pole socket (inches):  12.0

   CLIMATE DATA
              average daily rainfall (inches):   0.10
              average daily temperature (degs F):   50.0
              wind speeds (mph):  10.0
                   sustained average (mph):
                   gust speeds (mph):
              potential ice or snow load (psf): 2.0

     GENERAL TERRAIN AND FOLIAGE CLASSIFICATION:
          FIXED FACILITY SUPPORT SYSTEMS       PROJECT # TRIAL
                  Structure Geometry          REVISION # 0

    PLAN VIEW DATA

          length of structure (feet): 25.0
           bay spacings (measured along the structure length)

             2  @ 12.5    @         @         @         @
                @         @         @         @         @

          width of structure (feet): 20.0
          minimum pole spacing (feet): 5.0

    ELEVATION DATA
          exterior pole height (above ground feet): 10.0
          exterior poles guyed (Y/N): Y
          allowable netting sag (feet): 2.0
          allowable displacement error (inches):  0.10
```

```
              FIXED FACILITY SUPPORT SYSTEMS        PROJECT # TRIAL
              Support Member Material Screen       REVISION # 0

 INDICATE DESIRED SUPPORT MEMBER TO BE USED  (Y-Yes  N-No)

                      ALLOW   MODULUS X-SECT  MOMENT   UNIT
               LENGTH STRENGTH ELASTIC  AREA  INERTIA  PRICE
      MARK      (FT)   (KSI)   (10**6) (SQIN) (IN**4)  ($/FT)   DESCRIPTION
   N A0202P     20      30       10     1.75    .911     0     2 X 2 ALUMINUM
   N A0303P     20      30       10     2.75    3.49     0     3 X 3 ALUMINUM
   Y A0404P     20      50      29.0    6.36   12.35   0.80    4 X 4 ALUMINUM
   N W0404P     20       7      1.7      16      85      0     4 X 4 WOODEN PO
```

```
                          FIXED FACILITY SUPPORT SYSTEMS     PROJECT # TRIAL
                              Netting Material Screen       REVISION # 0

  INDICATE DESIRED NETTING TYPE TO BE USED  (Y-Yes  N-No)
           UNIT   NETTING   ALLOW    UNIT      DRAG    UNIT
           SIZE    THICK  STRENGTH  WEIGHT   COEFFIC   PRICE
   MARK   (SQFT)   (IN)    (KSI)   (#/SQFT)  (UNITS) ($/SQFT)   DESCRIPTION
N BF016N  0        0        0        0         0       0        BRUNSWICK 16-LB
N BF022N  0        0        0        0         0       0        BRUNSWICK 22-LB
Y SF050N  400.0    0.085    5.0      0.083     005     0.15     STANDARD DOD50
N TB035N  0        0        0        0         0       0        TELEDYNE BROWN
N UL060N  0        0        0        0         0       0        ULCAN 60-LB NET
```

```
                          FIXED FACILITY SUPPORT SYSTEMS     PROJECT # TRIAL
                          Tension Member Material Screen    REVISION # 0

  INDICATE DESIRED TENSION MEMBER TO BE USED  (Y-Yes  N-No)

            ALLOW    MODULUS  THERMAL X-SECTION  UNIT     UNIT
          STRENGTH   ELASTIC  COEFFIC   AREA    WEIGHT   PRICE
    MARK   (KIPS)   (10**6) (10**-6)   (SQIN)   (#/FT)   ($/FT)     DESCRIPTION
N GR416C     0         0        0         0        0        0       1/4 INCH GRASS
N GS216C   16.0       29       6.5      .012     0.40       0       1/8 IN AIRCRAFT
Y GS316C   17.5       29       6.5      .785     0.45     0.70     3/16 IN AIRCRAFT
N GS416C   20.0        0        0         0        0        0       1/4 IN AIRCRAT
N GS516C   22.0        0        0         0        0        0      5/16 IN AIRCRAFT
N GS616C   2.4         0        0         0        0        0       3/8 IN AIRCRAFT
N GS816C   4.2         0        0         0        0        0       1/2 IN AIRCRAFT
N NL416C     0         0        0         0        0        0       1/4 INCH NYLON
```

```
                              FIXED FACILITY SUPPORT SYSTEMS     PROJECT # TRIAL
                                Anchor Material Screen          REVISION # 0

  INDICATE DESIRED ANCHOR TO BE USED  (Y-Yes  N-No)

                      ALLOW    MODULUS   ROD      ANCHOR   UNIT
            LENGTH  STRENGTH  ELASTIC DIAMETER  DIAMETER  PRICE
    MARK    (INCH)   (KIPS)   (10**6)  (INCH)    (INCH)   ($/EA)    DESCRIPTION
 N T0146A    66       58        29       1.25      10      32.8    1 1/4 X 66 10-I
 N T0148A    96       58        29       1.25      10      39.6    1 1/4 X 96 10-I
 Y T0816A    66       36        29      1.00     8.0      20.5    1 X 66 8-INCH T
 N T2537A    96       58        29       1.25      14      59.6    1 1/4 X 96 14-I
 N T4345A    54       23        29       .75       4       11.0    3/4 X 54 4-INCH
 N T6346A    66       23        29       .75       6       13.7    3/4 X 66 6-INCH
```

```
                                      PROJECT:  TRIAL0
                                     REVISION:  0
                                     PROG VER:  01.01
                                     PROG NUM:     1

          STRUCTURAL SYSTEM SUPPORT MEMBER DEFLECTIONS

SUPPORT         TOP OF MEMBER DEFLECTION IN THE
 MEMBER      LENGTH DIMENSION       WIDTH DIMENSION
 NUMBER          (INCHES)               (INCHES)
    1            0.290962              -0.054643
    2            0.016011              -0.001484
    3            0.015925               0.003361
    4            0.291953               0.052228
    5            0.002508              -0.129863
    6           -0.001096               0.001685
    7           -0.001061              -0.001879
    8            0.002492               0.129701
    9           -0.289936              -0.051855
   10           -0.019525              -0.001472
   11           -0.019035               0.001489
   12           -0.291424               0.053115
```

```
                                                       PROJECT:  TRIAL0
                                                      REVISION:  0
                                                      PROG VER:  01.01
                                                      PROG NUM:     1

                      STRUCTURAL SYSTEM TENSION MEMBER LENGTHS

                       MIN DISTANCE    UNSTRESSED LENGTH
 FROM     TO            (INCHES)           (INCHES)           CABLE TYPE
   1       2             80.00              92.87        EXTERIOR  WIDTH
   2       3             80.00              92.87        EXTERIOR  WIDTH
   3       4             80.00              92.87        EXTERIOR  WIDTH
   5       6             80.00              92.87        INTERIOR  WIDTH
   6       7             80.00              92.87        INTERIOR  WIDTH
   7       8             80.00              92.87        INTERIOR  WIDTH
   9      10             80.00              92.87        EXTERIOR  WIDTH
  10      11             80.00              92.87        EXTERIOR  WIDTH
  11      12             80.00              92.87        EXTERIOR  WIDTH
   1       5            150.00             156.64        EXTERIOR  LENGTH
   5       9            150.00             156.64        EXTERIOR  LENGTH
   2       6            150.00             156.64        INTERIOR  LENGTH
   6      10            150.00             156.64        INTERIOR  LENGTH
   3       7            150.00             156.64        INTERIOR  LENGTH
   7      11            150.00             156.64        INTERIOR  LENGTH
   4       8            150.00             156.64        EXTERIOR  LENGTH
   8      12            150.00             156.64        EXTERIOR  LENGTH
   1       6            150.00             175.84        DIAGONAL
   2       5            150.00             175.84        DIAGONAL
   5      10            150.00             175.84        DIAGONAL
   6       9            150.00             175.84        DIAGONAL
   2       7            150.00             175.84        DIAGONAL
   3       6            150.00             175.84        DIAGONAL
   6      11            150.00             175.84        DIAGONAL
   7      10            150.00             175.84        DIAGONAL
   3       8            150.00             175.84        DIAGONAL
   4       7            150.00             175.84        DIAGONAL
   7      12            150.00             175.84        DIAGONAL
   8      11            150.00             175.84        DIAGONAL
```

```
                                                          PROJECT:  TRIAL0
                                                         REVISION.  0
                                                         PROG VER:  01.01
                                                         PROG NUM:     1

                  STRUCTURAL SYSTEM TENSION MEMBER LENGTHS

                    MIN DISTANCE   UNSTRESSED LENGTH
FROM      TO         (INCHES)          (INCHES)         CABLE TYPE
   1       1          169.71            169.71        HORZ       GUY
   5       5          169.71            169.71        HORZ       GUY
   9       9          169.71            169.71        HORZ       GUY
   1       1          169.71            169.71        VERT       GUY
   2       2          169.71            169.71        VERT       GUY
   3       3          169.71            169.71        VERT       GUY
   4       4          169.71            169.71        VERT       GUY
   4       4          169.71            169.71        HORZ       GUY
   8       8          169.71            169.71        HORZ       GUY
  12      12          169.71            169.71        HORZ       GUY
   9       9          169.71            169.71        VERT       GUY
  10      10          169.71            169.71        VERT       GUY
  11      11          169.71            169.71        VERT       GUY
  12      12          169.71            169.71        VERT       GUY
```

```
                                              PROJECT:  TRIAL0
                                             REVISION:  0
                                             PROG VER:  01.01
                                             PROG NUM:    1

                         FOUNDATION/FOOTING SYSTEM DESIGN

           LENGTH TO PREVENT OVERTURNING        4.000 FEET
           MOMENT APPLIED                    1847.966 FT-POUNDS
           ALLOWABLE MOMENT                  2145.840 FT-POUNDS
           LATERAL FORCE APPLIED              184.797 POUNDS
           ALLOWABLE LATERAL FORCE           1303.598 POUNDS


                     SELECTED ANCHORING SYSTEM DESIGN

    DIAMETER     DEPTH      CAPACITY     MAX LOAD     MAX STRENGTH
     (INCH)     (INCH)       (LBS)        (LBS)          (LBS)
   ________________________________________________________________

       8.00     66.00       2703.1        219.2        36000.0


                       STRUCTURAL MATERIALS SUMMARY

     QUANTITY    MARK    USE    DESCRIPTION      UNIT COST    TOTAL COST
          12.    A0404    P     4 X 4 ALUMINUM       0.80        192.00
          64.    SF050    N     STANDARD DOD 50      0.15       3840.00
        4200.    GS316    C     3/16 IN AIRCRAF      0.70       2940.00
        2376.    GS316    C     3/16 IN AIRCRAF      0.70       1663.20
          14.    T0816    A     1 X 66 8-INCH T     20.50        287.00

         NOTE:      1  CUT(S) REQUIRED ON EACH SUPPORT MEMBER
                    NETTING OVERHANG IS ONE-HALF STRUCTURE HEIGHT
                           ON ALL SIDES
                    NETTING OVERLAP IS 5.0 PERCENT ON ALL SIDES
```

Waterways Experiment Station Cataloging-in-Publication Data

Epps, James W.
Design of fixed-facility multispectral camouflage netting support systems / by James W. Epps, Marion W. Corey ; prepared for Department of the Army, US Army Corps of Engineers ; monitored by Environmental Laboratory, US Army Engineer Waterways Experiment Station.
236 p. : ill. ; 28 cm. — (Technical report ; EL-92-34)
Includes bibliographical references.
1. Camouflage (Military science) 2. Anchorage (Structural engineering) 3. Foundations — Military aspects. I. Corey, Marion W. II. United States. Army. Corps of Engineers. III. US Army Engineer Waterways Experiment Station. IV. Title. V. Series: Technical report (US Army Engineer Waterways Experiment Station) ; EL-92-34.
TA7 W34 no.EL-92-34